(2)

BBN Systems and Technologies
A Division of Bolt Beranek and Newman Inc.

BBN Report No. 7715

ARPA Order Number 6726
Contract Number: N00014-89-C-0008
Contract Duration: 1 May 1989 - 29 February 1992
Principal Investigator: J. Makhoul, (617)873-3332

Final Report

DEVELOPMENT OF A SPOKEN LANGUAGE SYSTEM

S. Austin, D. Ayuso, C. Barry, M. Bates, R. Bobrow, S. Boisen, D. Ellard, R. Ingria, F. Kubala, J. Makhoul, P. Peterson, P. Placeway, R. Schwartz, V. Shaked, D. Stallard

April 1992

bbn

BBN Report No. 7715

ARPA Order Number 6726
Contract Number: N00014-89-C-0008
Contract Duration: 1 May 1989 - 29 February 1992
Principal Investigator: J. Makhoul, (617)873-3332

Final Report

DEVELOPMENT OF A SPOKEN LANGUAGE SYSTEM

S. Austin, D. Ayuso, C. Barry, M. Bates, R. Bobrow, S. Boisen,
D. Ellard, R. Ingria, F. Kubala, J. Makhoul, P. Peterson,
P. Placeway, R. Schwartz, V. Shaked, D. Stallard

April 1992

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
April 1992

3. REPORT TYPE AND DATES COVERED
Final Report, 5/1/89 - 2/29/92

4. TITLE AND SUBTITLE
Development of a Spoken Language System

5. FUNDING NUMBERS
N00014-89-C-0008

6. AUTHOR(S) S. Austin, D. Ayuso, C. Barry, M. Bates, R. Bobrow, S. Boisen, D. Ellard, R. Ingria, F. Kubala, J. Makhoul, P. Peterson, P. Placeway, R. Schwartz, V. Shaked, D. Stallard

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BBN Systems and Technologies
10 Moulton Street
Cambridge, MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER
Report No. 7715

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Department of the Navy
Arlington, Virginia 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution of the document is unlimited. It may be released to the Clearinghouse, Dept. of Commerce, for sale to the general public.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)* This report describes the activities performed and the progress made in the development of HARC (Hear And Respond to Continuous speech), BBN's spoken language system, from May 1, 1989 to February 29, 1992. Significant progress has been made both in terms of speed of understanding and accuracy of understanding. New search algorithms BBN developed during this period increased the speed of HARC by more than three orders of magnitude. The result has been the first spoken language system that runs in real-time on an off-the-shelf workstation, with no additional hardware. Furthermore, real-time performance was achieved without losing understanding accuracy. In tests performed by the National Institute of Standards and Technology (NIST) on data collected from the DARPA Airline Travel Information System (ATIS) domain, the BBN HARC system had the best speech recognition and speech understanding performance. Another important milestone in this project has been the demonstration of HARC in a military logistical transportation planning application.

14. SUBJECT TERMS
Spoken Language System, Natural Language, Speech Recognition, N-Best Search, Unification Syntax and Semantics, Fragment Processing, Real-Time Recognition, Performance Evaluation

15. NUMBER OF PAGES
239

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	

Contents

Executive Summary

This report describes the activities performed and the progress made in the development of HARC (Hear And Respond to Continuous speech), BBN's spoken language system, from May 1, 1989 to February 29, 1992. Significant progress has been made both in terms of speed of understanding and accuracy of understanding. New search algorithms that we developed during this period have increased the speed of HARC by more than three orders of magnitude. The result has been the first spoken language system that runs in real-time on an off-the-shelf workstation, with no additional hardware. Furthermore, real-time performance was achieved without losing understanding accuracy. In tests performed by the National Institute of Standards and Technology (NIST) on data collected from the DARPA Airline Travel Information System (ATIS) domain, the BBN HARC system had the best speech recognition and speech understanding performance. Another important milestone in this project has been the demonstration of HARC in a military logistical transportation planning application.

Below is a list of some of the major accomplishments of this project:

- Extended the coverage and robustness of the syntactic and semantic components of our natural language system, DELPHI, already developed before the start of the project.
- Developed a discourse component, based on our previous work with natural language systems, to permit the use of DELPHI in an interactive spoken language system.
- Developed speech processing techniques to detect out-of-vocabulary words in users' utterances and to add them to the system.
- Developed a new paradigm, N-best, for the integration of knowledge sources (both in speech recognition and natural language understanding). The N-best paradigm has become the *de facto* standard for integrating knowledge sources in the DARPA community and elsewhere.
- Integrated DELPHI with our speech recognition system, BYBLOS, to form a complete spoken language system, HARC.

- Increased the speed of all components to develop a real-time spoken language system. The system runs in real-time on an off-the-shelf workstation, with no additional hardware.
- Ported HARC to a military logistical transportation planning domain.
- Created a videotape to illustrate these capabilities and some potential applications of spoken language technology.
- Participated in the DARPA cross-site data collection effort and delivered the stipulated speech data to NIST on time.
- Participated in all evaluation tests.

We now give a brief description of these highlights.

During this project we modified our natural language system, DELPHI, to deal with the range of utterances that might be used as the input to a spoken language system. Towards this end, we extended the coverage of the grammar and also developed alternative processing strategies to allow DELPHI to determine the meaning even of utterances that the system could not completely parse. We also developed a domain-independent discourse component and a domain-dependent plan-tracking component for the system, to allow DELPHI to be used in the natural dialogue situations that a spoken language system might be expected to encounter.

We continued our work in developing new speech recognition algorithms. In particular, we developed techniques that allow BYBLOS to recognize out-of-vocabulary words and to add them to the system's dictionary so that the system can recognize them in future usage. This capability was demonstrated in the DARPA Resource Management task domain.

One of the major developments has been the introduction of the N-best paradigm for integrating knowledge sources in speech recognition and understanding. The idea is that an initial speech recognition pass produces, not only the top scoring sentence, but also the top N (or N-best) sentences, where N is typically less than 100. The N-best list is then rescored with other, more expensive knowledge sources, and the list is reordered. For example, we integrate statistical trigram grammars in this way, as well as cross-word rescoring. We also use the N-best paradigm to integrate our speech recognition and natural language components. The major benefits of the N-best paradigm are significant increases in speed of processing and improvement in overall recognition and understanding performance. The N-best algorithm, which was first presented in October 1989, has become the community wide *de facto* standard for integration of knowledge sources, especially the integration of speech and natural language.

The integration strategy for our speech recognition and natural language components adopted for HARC, our spoken language system, is the following. The speech recognition component produces a list of N-best hypotheses, which are passed to the natural language processor. (For this part of our system, we have found a value of N=5 to give best understanding results.) The natural language system processes these hypotheses until a hypothesis which produces an answer (a database retrieval) is encountered or until the list of hypotheses is exhausted. The complete HARC system has been demonstrated in the ATIS and DART domains (see below).

We participated in all the evaluations sponsored by DARPA and performed by NIST. In the most recent (February 1992) evaluation in the ATIS domain, our BYBLOS system had the best speech recognition performance and our HARC system had the best speech understanding performance of all the sites participating in the evaluation.

A major milestone during the course of the project was the demonstration of our spoken language system HARC in a military logistical planning application: DART (Dynamic Analysis Replanning Tool). In this demonstration, speech was used to replace large numbers of computer mouse clicks to retrieve information from a database of logistics information. A videotape demonstrating this application of spoken language technology was made and was shown at the 1991 DARPA Speech and Natural Language Workshop, as well as to interested government personnel.

The basic evaluation methodology used in all the cross-site evaluations involving natural language was proposed and initially developed by BBN. Continuing the tradition of objective evaluations begun for speech recognition systems, we proposed an evaluation technique in which systems utilizing a natural language "understanding" component, whether with text or speech input, would be evaluated in terms of the output answers they produced from a fixed, communally available database. We created the CAS (Common Answer Specification) format used for such answers and created the first software program for comparing hypothesis and reference answers. We participated in the development of evaluation strategies for pairs of related sentences, and, ultimately, for evaluating "full sessions".

As part of the cross-site SLS data collection effort that began in June of 1991, we constructed a "Wizard" data collection system that incorporated DELPHI and BYBLOS. We collected over 2200 sentences in all, fulfilling the SLS community's agreement to collect this amount of data by the end of August 1991. (We were the only site to actually meet this deadline.)

The crowning achievement of this project was the development of a complete spoken language system working in real-time on an off-the-shelf workstation. New search algorithms that resulted in more than three orders of magnitude increase in computing speed of our combined HARC system allowed this accomplishment to take place. The system can perform real-time speech recognition of continuous speech with a 1000-word vocabulary, give an N-best list as well, rescore with a trigram grammar for higher accuracy, and perform natural language understanding as well, all completely in software. The system has been demonstrated on the Indigo workstation Silicon Graphics Inc. and the SUN SparcStation 2. The real-time system uses the A/D internal to the workstation for speech input. Being a software-only system, our real-time system can be ported easily to other applications and other platforms.

Ours is currently the only system in the DARPA community that can run in real-time in the ATIS domain and is also the system with the best understanding performance.

Report Outline

The structure of the rest of this report is as follows. Chapter 1 provides a general overview of the current architecture of HARC and details the changes from the version described in [24]. Chapter 2 describes the grammar formalism, concentrating on the aspects of our approach to syntax that provide for maximum flexibility and coverage. Chapter 3 describes the current semantic processing framework, and the way in which the syntactic phenomena described in Chapter 2 are interpreted by the semantic component. Chapter 4 describes the parsing and fallback strategies which we have adopted to increase speed and robustness of syntactic processing. Chapter 5 describes the discourse and task-tracking components we implemented during this project. Chapter 6 describes the N-best algorithm and its use in the integration strategy for speech and natural language that we have now adopted. Chapter 7 describes our development of a real-time speech recognition system. Chapter 8 describes our development of techniques to detect new words in speech and to allow the user to add them to the system. Chapter 9 describes our co-operative efforts with other research sites. Chapter 10 describes our application of spoken language system technology to a military domain. Chapter 11 describes our lead role in producing an evaluation methodology for natural language and spoken language systems and our participation in the common data collection effort. Chapter 12 gives a list of the papers published by the members of the project over the course of the project, as well as oral presentations and conference submissions that have been accepted for presentation. The Bibliography contains references pertinent to this work.

Chapter 1

System Overview

This chapter provides a general overview of HARC, the BBN Spoken Language System. The general architecture of HARC is shown in Figure 1.1. The user presents a spoken utterance to BYBLOS, our continuous speech recognition system, which produces a list of N-best possible hypotheses as to the sentence spoken. This N-best list is passed to DELPHI, our natural language understanding system, which processes the N-best until it finds a hypothesis that produces a data base retrieval command. This is then passed to the application database, producing an answer that is displayed to the user.

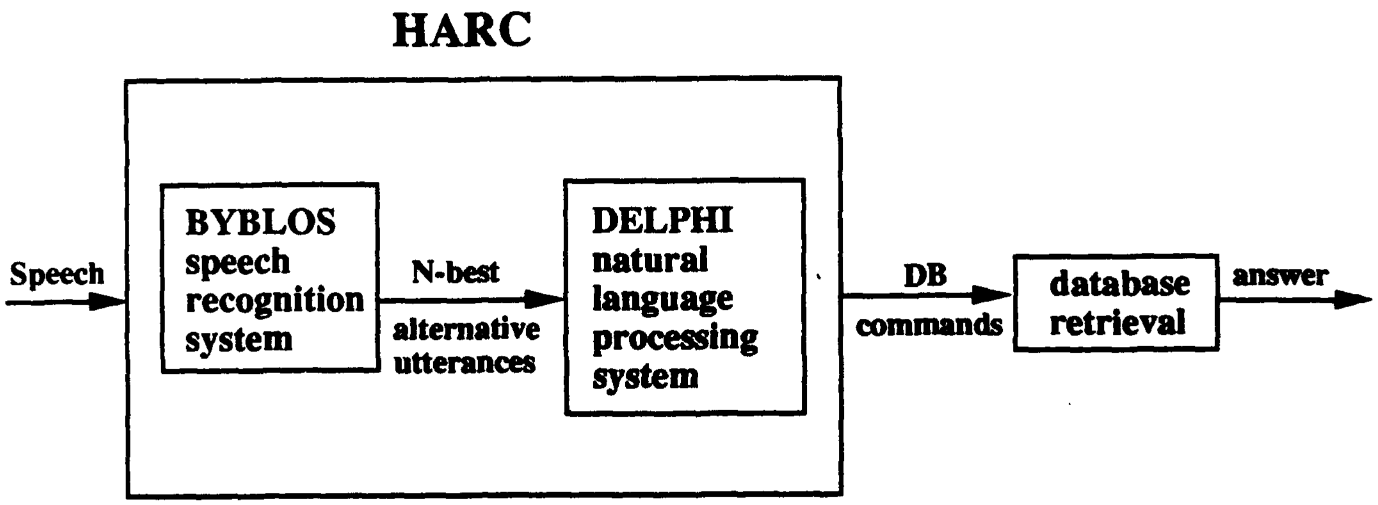

HARC
Speech
BYBLOS speech recognition system
N-best
alternative utterances
DELPHI natural language processing system
DB
commands
database retrieval
answer

Figure 1.1: The Architecture of HARC

This chapter describes, in broad terms, the improvements to DELPHI, our natural language processing system (Section 1.1), BYBLOS, our speech recognition system (Section 1.2), and the integration of these two components into a spoken language system (Section

1.3), from the version of the system described in [24] [25]. The chapter concludes with a summary of implementation specifications (Section 1.4).

1.1 DELPHI (Natural Language Component)

During the predecessor to this contract, we produced the initial version of the DELPHI natural language understanding system. During the course of that project, we designed and implemented a grammar formalism, adopted an all-paths parsing algorithm, and implemented a Montague-style [65] semantics, in which each rule of the grammar had an associated semantic interpretation rule. Owing to the fairly simple nature of the application domain, that system had a purely sentential semantics, with no discourse capabilities.

During the course of this project, we have made a number of changes to DELPHI:

- We have reworked the grammar rules and grammar formalism to produce a grammar that can be more efficiently parsed and interpreted.
- We have integrated semantics with syntax, to improve system performance and robustness.
- We have added a discourse component to the system, to allow it to work in real application domains.

Our main research goal during this project has been to take the tools currently available to grammar developers (such as unification, efficient parsing algorithms, etc.) and to integrate them into the most efficient and robust overall system possible. We have tried to not rely on any one mechanism too heavily but rather to deploy each where it functions optimally.

The following sections give a rough overview of our changes to the grammar, semantics, and parser, and the current implementation of our discourse component.

1.1.1 The Syntactic Component: the Grammar Formalism

The original DELPHI grammar formalism was a relatively "stripped down" version of the complex-feature based grammar formalisms that are common today; see [87] for an overview of many such formalisms. It was, in effect, an implementation in LISP of the common Definite Clause Grammar (DCG) formalism [74] available in Prolog. In particular, it did not support the use of the relatively sophisticated mechanisms that are part of other formalisms, such as feature disjunction, feature negation, metarules, optional arguments, and the use of attribute-value pairs, rather than positional arguments. In general, we have found this Spartan character to actually be of great use in writing a large grammar and have mostly maintained it. However, we have introduced a limited use of feature disjunction, described in Section 4.1.1. In addition to this change, there have been several other major changes in the grammar proper.

We have reduced the size of many subparts of the grammar by adopting a recursive structure for various categories. For example, in earlier versions of the grammar, there were many rules for different expansions of the common noun phrase. The noun phrase grammar now consists of a CORE-NP constituent, which spans the portion of the noun phrase from the determiner to its head (e.g. "which flights", "how many airlines", etc), a unit production that expands NP as CORE-NP, and recursive expansions of NP, each adding a (typically single) constituent that may be interpreted as a complement or adjunct to the noun phrase, such as relative clauses ("that leave on Sunday"), prepositional phrases ("from Boston"), participial phrases ("leaving after 6 PM"), etc. We discuss this aspect of the grammar in Section 2.3.

To aid in efficient parsing, we have split apart various categories that were formerly unitary. For example, the earlier grammar contained a single S (Sentence) category that was used for main clauses (*"What flights do you have from Boston to Baltimore?"*), relative clauses ("a flight *that arrives around eleven o'clock in the morning*"), and other subordinate clauses ("I'd like to know *what flights United Airline has from Dallas to San Francisco.*") However, these types of clauses have very different syntactic properties and treating them as a single category did not permit the optimal parsing strategy. We have since split S into three different categories. More details of similar sorts of category division are discussed in Section 4.1.2.

The earlier DELPHI grammar made sparing use of constraint relations. These are grammar rules or constituents thereof that do not derive elements in the linguistic input but rather solve equations constraining the well-formedness of the parse produced. An example of such a constraint relation from the earlier grammar was P-MIN, which computed the person of a conjoined noun phrase (for the purposes of subject verb agreement). In the current project, we have made extensive use of constraint relations, for a number of purposes, both syntactic and semantic. These are discussed in more detail in Sections 2.2 and 3.6. We have also introduced a mechanism to compile constraint relations into executable code, so that our grammar formalism now contains declarative rules with attached procedures.

Finally, we have reduced the size of the grammar by moving terminal elements like the articles ("a", "the", etc), prepositions, ("in", "on", "at", etc), and other grammatical formatives (loosely, form words rather than content words) which were previously introduced directly in the grammar, into the lexicon.

The current DELPHI grammar contains 453 rules to handle the general syntactic constructions of English.

1.1.2 The Semantic Component

In the earlier version of the DELPHI system, a single semantic rule was assigned to each grammar rule. Semantic processing did not take place during parsing. Rather, the parser would return all parses syntactically compatible with the linguistic input. A post-processor would then process each parse, traversing the parse tree in a recursive descent, and filtering out all parses that were not semantically coherent. This generate and test paradigm, given the wide range of syntactic analyses compatible with a linguistic input without semantic constraints, was not optimal. During this project, we have worked to establish the best link-up between syntax and semantics. Our current architecture applies semantic type constraints during parsing (for example, in the ATIS domain, a Noun Phrase object of the verb "leave" must be either an airport or a city) and constructs larger semantic formulae out of the interpretations of syntactic subcomponents only at levels where all the elements that contribute to that formula are present. For example, in interpreting the utterance "Does United Airline have any flights from Dallas to San Francisco?" it is only at the level of the entire sentence that the semantic interpretations of the Noun Phrase "United Airline", the Verb "have" and "any flights from Dallas to San Francisco" are composed. Crucially, there is no partial interpretation of the Verb Phrase "have any flights from Dallas to San Francisco", as there would be in most unification systems. The evolution of our approach to semantic interpretation is described in Chapter 3.

1.1.3 Parsing

The parsing algorithm originally used to parse the DELPHI grammar was based on the Graham, Harrison, Ruzzo (GHR) algorithm [39], itself an extension of the familiar Cocke-Kasami-Younger (CKY) algorithm for context-free grammars. Our algorithm was, in turn, an extension of the GHR algorithm to support complex feature-based grammars—the original GHR algorithm only handles true context-free grammars. In our earlier implementation of the GHR algorithm, we omitted the prediction portion of the algorithm: this is a mechanism that utilizes the current initial syntactic context to try to predict the (most likely) following syntactic constituent(s). The reason for the omission is that the notion of syntactic constituent in a feature based grammar is quite complex, encompassing not only category label, but syntactic features as well (for example, at the level of category label, a singular NP like "the flight" and a plural NP like "flights" are the same type of constituent; however if the number feature is included, they are distinct). In earlier work, we discovered that prediction based purely on category label was too weak to improve parsing performance, while prediction using all syntactic features was too computationally expensive. However, the division of categories such as S into multiple categories allows us to use prediction by category label alone to good effect. This is discussed in more detail in Section 4.1.2.

While this modification to the GHR algorithm increased processing speed, we still found the processing time to be too slow for the requirements of a real-time spoken language system. Therefore, to overcome the limitations of the GHR algorithm, we created a second algorithm, which utilizes the same parse tables and chart structures as our implementation of the GHR algorithm, and which work in a best first manner, using statistical grammar rule scores derived from a training corpus. The agenda mechanism we have used to implement this best-first search is described in Section 4.2. This best-first algorithm is what we normally use in our current system, but our GHR parser is also available for situations in which the best-first algorithm is not appropriate.

Finally, we have introduced fallback processing into our use of grammatical information. Given the presence of false-starts, idiolectal variation, and other extra-grammatical elements in spoken utterances, it is virtually impossible to cover all spoken input with a restrictive grammar. The details of the mechanisms we have implemented are presented in Section 4.3.

1.1.4 Discourse

The application domain of the initial version of the DELPHI System was Resource Management. Since most of the queries in this domain were simple database retrieval queries or commands to the display system, there was little need for discourse processing. However, the full range of applications of spoken language systems clearly requires discourse capabilities. During the current project we have developed a discourse component that utilizes domain independent mechanisms for pronoun interpretation based on earlier work with the Janus system. We have also developed a plan tracker that utilizes task and domain specific information to guide discourse interpretation. Our work in this area is described in Chapter 5.

1.2 BYBLOS (Speech Recognition Component)

The basic speech recognition technology used in this contract was developed under a separate contract, and under that contract we continue to improve the accuracy of our speech recognition system, BYBLOS.

Under this contract, our speech recognition work focussed on developing new, more efficient methods for integrating speech recognition with natural language, real-time recognition on commercially available hardware, developing a real-time demonstration, and building several specific components for a useful speech recognition system. Specifically:

- We reduced the computation needed for speech recognition by taking advantage of the back-off structure of statistical n-gram grammars. We also greatly increased the efficiency of the speech recognition search to the point where it can run in real time on an SGI 4D/35 (which is about 3-4 times faster than a SUN 4/280) with vocabularies of over 1,000 words. Thus no special purpose hardware is required.
- We developed the N-Best Paradigm for integrating speech recognition and natural language. We also developed the first exact algorithm for finding the N-Best sentence hypotheses, and then developed several more efficient approximate algorithms.
- We applied our Forward-Backward Search technique (developed in the previous contract for integrating Multiple Knowledge Sources) to speed up the N-Best search by an additional factor of 40.
- We developed the first algorithm for detecting when a speaker says a word outside the vocabulary. We also implemented a method that allows a naive user to add new words to the dictionary.

1.2.1 Recognition Speed

In the previous contract for integrating speech recognition with natural language, the speech recognition component produced a dense lattice of word hypotheses, which were then searched (parsed) by the natural language component. At the beginning of this contract a 1-best search required about 100 times real time to run on a SUN 4/280. Creating the word lattice required about 1,000 times real time. We clearly needed another approach to increase the speed of our speech recognition.

In a previous contract for using parallel processors for speech recognition we showed that it was possible to use a large parallel processor to speed up the recognition search. In particular, we used a Butterfly^{TM} parallel processor with 98 nodes to speed up a simple search by a factor of 77. In addition, we used 32 nodes to speed up a more complex grammar search by a factor of 16. These improvements in recognition speed, while very impressive, were not sufficient for our purposes and did not justify the significant cost of the parallel processors or the additional time taken to program a parallel machine.

A separate solution for increased speed was being investigated at SRI and UC Berkeley, namely, the design of special-purpose VLSI chips for real-time speech recognition. That effort was started in 1988 but had not achieved its objective by the end of this contract in February 1992.

At BBN, we took a very different point of view. Our primary goal in terms of recognition speed was to eliminate the need for special purpose or parallel hardware for the recognition search. This required that we greatly reduce the time needed for the basic recognition search algorithmically. The type of language model we use most often is a fully-connected statistical n-gram grammar, which represents the probability of each word given the previous one or two words. We reduced the computation required for this kind of grammar significantly by taking advantage of the back-off structure of this general type of grammar. In particular, in a bigram grammar, instead of having a separate grammar transition for each pair of words, we realized that we only needed to include those transitions corresponding to word pairs that were actually observed in the training. All other transitions are accommodated by a transition with a back-off probability from the end of each word to a "unigram node" from which there is a unigram transition to the beginning of each word. This simplification greatly reduced the computation needed with no change in accuracy.

In addition to the above approach, we also reimplemented the recognition program, using good software engineering practices, in order to make it more efficient. The net result is that a 1-Best recognition with over 1,000 words can now be performed in real time on a workstation. This means that as soon as a speaker has stopped speaking, the program will print out the most likely sentence hypothesis.

Finally, we developed a real-time demonstration system using commercially available hardware. In particular, we used a workstation for the recognition search, with a readily available signal processing board to perform the front end signal analysis and vector quantization. This demonstration, which was given in August 1991, was, to our knowledge, the first real-time demonstration of continuous speech recognition with 1,000 words (without excessive loss of accuracy). The fact that it was performed on commercially available hardware was therefore even more astounding.

Subsequently, a separate effort at BBN to commercialize speech recognition technology developed a much faster version of the front end signal processing that was able to run completely on the workstation in a fraction of real time. Therefore, by using a workstation with a built-in A/D capability (like the SGI 4D/35 or the SUN SparcStation 2), we now have real-time recognition operating completely on a standard workstation. This system was demonstrated at the February 1992 DARPA Speech and Natural Language Workshop.

1.2.2 The N-Best Paradigm and Algorithms for Finding N-Best Hypotheses

We observed that a tight coupling of several knowledge sources often resulted in very large computation. We postulated that a more efficient integration of several knowledge sources might be achieved by judiciously ordering the knowledge sources so that those that were powerful and yet inexpensive to apply were used first.

We developed the N-Best Paradigm in which the BYBLOS system now produces not one, but several (the N Best) alternative sentence hypotheses using a discrete HMM with a bigram grammar. These hypotheses are first rescored using more detailed speech knowledge sources, including between-word coarticulation models, semi-continuous density HMMs and trigram statistical language models. Then, the reordered list is given to the DELPHI natural language system, which searches for the highest scoring hypothesis that is meaningful and results in a valid answer.

We considered and developed several algorithms for finding the N-Best hypotheses. These included the Exact Sentence-Dependent N-Best algorithm, the approximate Word-Dependent N-Best algorithm, and the very fast Lattice N-Best algorithm. The Exact algorithm can be shown to guarantee that we can find the correct "forward probability" for all of the hypotheses that are within a threshold of the most likely hypothesis. The computation required is only linear in the number of hypotheses found.

The approximate N-Best algorithms were developed in an attempt to reduce the computation needed without sacrificing accuracy. We chose to use the Word-Dependent N-Best algorithm, because we found that it was empirically as accurate as the exact algorithm, and yet does not require as much computation.

1.2.3 Forward-Backward Search Algorithm

Even though the new Word-Dependent N-Best algorithm was quite fast, it was not fast enough to be performed in real time. We needed a way to speed it up by more than an order of magnitude. In the previous contract on combining Multiple Knowledge Sources we had developed the Forward-Backward Search Algorithm. In this method, we use a simplified forward search to speed up the computation of a more complex backward search. We applied the Forward-Backward Search Algorithm to speed up the computation of the N-Best search by a factor of 40 with no loss in accuracy. This meant that the N-Best computation, which only begins after the speaker has stopped, is completed in a small fraction of real time, or typically a delay of one second. We felt that this delay was not excessive for most applications.

1.2.4 Detecting and Adding New Words

One problem in a large vocabulary speech recognition system is that the user can not remember which words are in the vocabulary. Often, a user will say a word outside of the vocabulary resulting in a recognition error. However, the user will simply try to say the sentence again, not realizing that the word is not in the vocabulary. We felt that it would be quite desirable for the system to warn the user when a new word has been spoken, even though the system has no idea ahead of time of what new words may be spoken.

We developed a technique for detecting out-of-vocabulary words. The technique is based on having a general alternate model for new words. The model, which is constructed from a network of phoneme models, is designed to match any word reasonably well. However, it does not match existing words as well as the specific models for those words. The technique was tested using the DARPA Resource Management corpus. When operating in speaker-dependent mode, we were able to detect over 70% of the new words spoken, while falsely detecting new words in only 1% of the sentences. In speaker-independent mode the detection rate dropped to 50% with a false acceptance rate of 2%.

Once the system has detected a new word, the user needs a way to add it, even though they do not know how to create phonetic spellings. The system asks the user to type the word and say it once. It then uses the combination of the typed spelling and the spoken utterance to determine the most likely phonetic spelling of the new word so that it can be incorporated in the system.

1.3 Speech and Natural Language Integration

In our initial integration of Speech Recognition and Natural Language Processing technologies, we used a word lattice as the data structure to interface the two components. The speech component would produce a lattice of all the word hypotheses found and natural language would find all syntactically permissible and semantically acceptable utterance hypotheses covering the speech input. This was a computationally expensive procedure.

During this project we have moved to using the N-best hypotheses lists produced by speech as the data structure for interfacing the two components. DELPHI processes these hypotheses until either a hypothesis which produces an answer (a database retrieval) is encountered or until the number of hypotheses processed crosses a threshold, which can be varied. We have experimented with a number of variations on this strategy, and have found that a threshold of 5 produces the best results. We have also experimented with optimal ways of incorporating our fallback strategies. We have found that a two pass scheme, in which the N-best hypotheses are first processed with fallback processing disabled, and in which fallback is utilized as a second pass only if no answer is obtained, is superior to architectures which either totally omit fallback processing or use the system with fallback processing as the first pass. More details are given in Section 6.4.

1.4 Implementation

DELPHI is implemented in Common LISP. The earliest versions of DELPHI were developed on Symbolics 3600-class LISP Machines and TI Explorers. There are currently versions on Sun RISC-based workstations and Silicon Graphics machines running Lucid Common LISP under UNIX. The main computational core of the system is machine-independent; the identical code runs on all the systems named. A device-independent loader file allows the system code to be run on all these machines; it is conditionalized to handle input/output differences among them. During this project, we also developed a "configuration" mechanism that parameterizes the load procedure for the DELPHI system to load in collections of data files (lexicon and semantics) for different domains. This has allowed us to switch quickly from one version of the ATIS database to another, as well as to configurations for other domains.

BYBLOS is implemented in C and runs on UNIX workstations. HARC runs on UNIX machines, using standard UNIX facilities to interface DELPHI and BYBLOS. The user display for HARC is written using X Windows.

Chapter 2

Syntax and Grammar

In this section, we briefly describe the basic grammar formalism used in DELPHI and the changes that have been made to the formalism and to the grammar during the course of this project. One of the major changes made to the grammar is the wide-spread use of a device which is frequently made use of in Definite Clause Grammar work, but which has less often been utilized in the work on unification grammar that has appeared more recently ([87] [66]). This is the right-hand side "constraint relation" that does not derive any constituent of the utterance, but rather serves to constrain in various ways the feature assignments of those rule elements which do. Constraint relation elements are thus to be distinguished from other non-terminals which derive empty constituents, such as gaps or zero morphemes. We show their utility for syntactic purposes in this chapter and discuss their uses for semantic interpretation in Section 3.6.

Another major change is the replacement of many idiosyncratic rules for the expansion of different categories with a recursive "layer of the onion" type structure which has allowed us to reduce the number of rules for the lexical categories NP, VP, and AP, while retaining or even increasing coverage.

We have also split what were formerly single categories into multiple distinct categories in order to make better use of prediction in parsing. Since the reasons for these splits are closely tied to their benefits to parsing, we merely mention these changes here and describe them in detail in Section 4.1.2, which discusses prediction.

Finally, we have introduced relational labels into the right hand side of grammar rules. Labels include the traditional "Head" (for example, in the Noun Phrase "flights from Boston to Denver", "flights" is the head), grammatical relations such as "Subject", "Direct Object", etc, and ad hoc labels for relations for which there exist no traditional names. The main utility of such labels is that they provide a more uniform and simpler interface to semantics. We introduce them briefly here in Section 2.4 and discuss their role in semantic interpretation in Section 3.5. The use of relational labels is more than just a convenience. In our current view of the relation between syntax and semantics, semantic interpretation is seen as a process operating on a sequence of messages characterizing local grammatical relations among phrases, rather than as a recursive tree walk over a globally complete and coherent parse tree. The combination of incremental semantic interpretation and statistical control of the parsing process (described in Section 4.2) makes it feasible to reconstruct local grammatical relations with substantial accuracy, even when a global parse cannot be obtained. Grammatical relations provide the interface between syntactic processing and semantic interpretation, and standard global parsing is viewed as merely one way to obtain evidence for the existence of such grammatical relations in an input string. This focus on grammatical relations has lead to substantial simplification of both grammar and semantic rules, and will facilitate our ultimate aim of acquiring syntactic, semantic and lexical knowledge by largely automatic means.

Note that all the rules presented in Sections 2.1–2.3 are in their current forms. However, since we only introduce the use of relational labels in Section 2.4, this being our most radical departure from standard complex feature based grammar formalisms, we have suppressed the relational labels in the rules in these earlier sections for the sake of exposition.

We begin our discussion of grammatical changes by first reviewing the basics of our grammar formalism; full details appear in [24].

2.1 The Grammar Formalism

DELPHI uses a grammar formalism based on annotated phrase structure rules. While it is in the general tradition of augmented phrase structure grammars, its immediate inspiration is Definite Clause Grammars (DCGs) [74]. In such grammars, rules are made up of elements that are not atomic categories but, rather, are complex symbols consisting of a category label and feature specifications. Features (also called arguments) may be either constants—indicated by lists in DELPHI—or variables—indicated by symbols with a leading colon (:). Identity of variables in the different elements of a rule is used to enforce agreement in the feature indicated by the variable. A (somewhat simplified) example of an actual grammar rule, which introduces conjoined Noun Phrases, is shown in Figure 2.1.

This rule states that an NP can consist of an NP, followed by a conjunction (CONJ), followed by another NP. Here, the identity of the `:POSS` feature, which indicates whether a Noun Phrase is possessive ("Delta's") or not ("Delta"), on the lefthand NP and in the two NPs on the righthand, requires that all conjuncts be either possessive or non-possessive,

```
(NP (AGR :P (PLURAL)) (REALNP :REALZ) :POSS :WH ...:DET-CLASS ...)
→
(NP (AGR :PERSONX :NUMX) (REALNP :REALX) :POSS :WH ...:DET-CLASSX ...)
(CONJ ...)
(NP (AGR :PERSONY :NUMY) (REALNP :REALY) :POSS :WH ...:DET-CLASSY ...)
{P-MIN :PERSONX :PERSONY :P}
{NPTYPE-FILTER :REALX :REALY :REALZ}
```

Figure 2.1: A Grammar Rule for Conjoined Noun Phrases

but does not allow a mixed conjunction. Similarly, the identity of **:WH**, which indicates whether a Noun Phrase contains a question element ("who", "which flights") or not ("them", "the flights") in the lefthand NP and the two conjuncts requires a conjoined NP to consist entirely of question or non-question Noun Phrases. On the other hand, the appearance of the distinct variables **:DET-CLASS**, **:DET-CLASSX** and **:DET-CLASSY** on the left and right hand sides of the rule, allows Noun Phrases with different classes of determiners to be conjoined.

P-MIN and NPTYPE-FILTER are constraint relations whose function will be discussed in the next section. We introduce here the convention of indicating constraint relations with curly braces ({}).

This formalism, like DCGs in general, maintains the term unification practice of using functors with obligatory, positional arguments, rather than functorless feature structures with optional, labelled arguments, as in much recent work, such as GPSG [37], LFG [28], and PATR-II [88]. There are several reasons for this. First of all, we find the functor very useful as a way of indicating just what features are allowed in a given structure (and it is interesting to note attempts to restore it for just this purpose in [64].) Second, it is relatively straightforward to have a simple syntactic checker that ensures that all grammar rules are well-formed. In a grammar as large as DELPHI's, having the ability to automatically make sure that all rules are well-formed is no small advantage. Finally, argument labels contribute their own clutter to the rule. They would thus seem to be a notational win only if more than half the features of a given element are "don't-cares", i.e. are not required to agree with features elsewhere in the rule. In our grammar, however, we find on average that only about 3% of the feature slots in rules are "don't-cares".

Several of the formalisms mentioned also contain other advanced mechanisms, such as feature disjunction, feature negation, metarules, and optional arguments. The occasions in which we have found the use of negation of various sorts useful are handled by the compiled constraint relations discussed in Section 2.2.1. Metarules would have been useful in the earlier version of the system, which used traditional subcategorization frames and, therefore, needed to "compile out" all the possible variants of each subcategorization frame into separate rules. However, the mapping unit strategy for subcategorization which we discuss in Section 3.4, eliminates the need for this. The mapping unit approach to subcategorization combined with the recursive structure of constituents discussed in Section 2.3 handles the phenomena normally treated by an optional arguments mechanism. We have found it useful to add a limited form of feature disjunction, which is described in Section 4.1.1.

2.2 Syntactic Constraint Analyses

While constraint relations have long been used in DCG-based work to treat, among other things, problems of quantifier scoping and the interaction of quantifier scope and anaphora [73] [75], we have found that there are other, more low-level issues of syntax and lexical semantics for which constraint relations are either conceptually useful or formally necessary, even in unification formalisms which allow full disjunction across features, such as [52] [55] [54] [49]. In particular, we discuss the constraints imposed by subgrammars on conjunction and relaxation of subject verb agreement in this section, and describe the use of constraint relations for semantic interpretation in Section 3.6.

Restrictions on agreement between features sometimes depend on the values of still other features. For example, English noun phrase conjunctions require that if one of the conjuncts belongs to a particular subgrammar (date, time etc.) then the other conjunct(s) must also belong to that subgrammar, but if neither of the conjuncts belongs to a subgrammar, there is no restriction. This can be done by including in the NP conjunction rule a constraint relation which provides a case-analyzing effect that could not be provided merely by unifying the variables ranging over the semantic type. This function is carried out by NPTYPE-FILTER, which we introduced above. This relation takes as its arguments the NPTYPE feature of the first conjunct, the NPTYPE of the second conjunct, and the NPTYPE feature of the conjunction as a whole. This relation, then, can be thought of as constraining the possible triples of these values.[1] We present four instances of this rule for illustration in Figure 2.2, where "$\rightarrow \emptyset$" indicates production of the empty string.

The first two rules require that if either of the conjuncts of a conjoined NP belongs to the TIMENP or DATENP subgrammar, than the other conjunct must belong to that subgrammar, as well. The last two rules allow ordinary pronominal and non-pronominal NPs to conjoin freely.[2] In this example, note that while the first two rules could be collapsed into a single rule utilizing unifying variables, the third and fourth cannot.

[1]Alternatively, it can be viewed as computing a value for the NPTYPE feature of the conjunction from the values of the individual conjuncts.

[2]They also set the NPTYPE of the conjunction to being non-pronominal, since it cannot function as a pronoun.

```
(NPTYPE-FILTER (NONUNITNP (-PRO (TIMENP)))
               (NONUNITNP (-PRO (TIMENP)))
               (NONUNITNP (-PRO (TIMENP))))
               → ∅
(NPTYPE-FILTER (NONUNITNP (-PRO (DATENP)))
               (NONUNITNP (-PRO (DATENP)))
               (NONUNITNP (-PRO (DATENP))))
               → ∅
(NPTYPE-FILTER (NONUNITNP (-PRO (MISCNP)))
               (NONUNITNP (+PRO :PRO-TYPE))
               (NONUNITNP (-PRO (MISCNP))))
               → ∅
(NPTYPE-FILTER (NONUNITNP (+PRO :PRO-TYPE))
               (NONUNITNP (-PRO (MISCNP)))
               (NONUNITNP (-PRO (MISCNP))))
               → ∅
```

Figure 2.2: A Sample Constraint Relation 1: NPTYPE-FILTER

Another use of constraint relations is to "compute" a value from the feature values of the relevant constituents, rather than requiring identity of features. For example, in NP conjunction with "and" in English, the person of the conjoined NP is first person if any of the conjuncts is first person, second person, if any of the conjuncts is second person and none is first person, and third person if all the conjuncts are third person.[3] This can be easily handled by the P-MIN constraint relation, which takes as its arguments the PERSON feature of the first conjunct, the PERSON of the second conjunct, and the PERSON feature of the conjunction as a whole. It has the solutions shown in Figure 2.3.

```
(P-MIN (1ST) :P (1ST)) → ∅
(P-MIN (3RD) :P :P) → ∅
(P-MIN (2ND) (1ST) (1ST)) → ∅
(P-MIN (2ND) (3RD) (2ND)) → ∅
(P-MIN (2ND) (2ND) (2ND)) → ∅
```

Figure 2.3: A Sample Constraint Relation 2: P-MIN

Still another case in which constraint relations provide a kind of flexibility greater than that available using standard unification is to allow the grammar to express "degrees of grammaticality". For example, in standard written English, it is common for a verb to agree in number with its subject.[4]

[3]Karttunen [52] handles such cases with a process of generalization, rather than unification.

[4]All of the examples in this discussion of subject-verb agreement are taken from the ATIS0 and ATIS1

What do the restrictions represent?
What does restriction VU/1 mean?
What do the transport codes AL and R mean?

However, in spoken English, conjoined noun phrases sometimes appear with singular agreement on the verb.

What does RETURN MIN and RETURN MAX mean?
What does class B and class Y mean?

In still looser speech, agreement disappears even with non-conjoined subject noun phrases:

List all the airlines that flies from Dallas to Boston nonstop.

These facts can be handled by modifying the standard sentence rule:

```
(ROOT-S ...)
→
(NP :AGR ...:CONJC ...)
(VP :AGR ...)
```

to the following:

```
(ROOT-S ...)
→
(NP :AGR ...:CONJC ...)
(VP :AGRX ...)
{SUBJECT-VERB-AGREEMENT :AGR :AGRX :CONJC}
```

and adding the solutions for SUBJECT-VERB-AGREEMENT shown in Figure 2.4. This is a constraint relation that takes as its arguments the agreement feature of the sentence's subject NP, the agreement feature of the sentence's VP, and the conjunction feature of the subject NP—which indicates whether it is a conjunction or not.

corpora collected by TI and distributed by NIST.

```
(SUBJECT-VERB-AGREEMENT (AGR :P :N)
                        (AGR :P :N)
                        :CONJ)
                        → ∅
(SUBJECT-VERB-AGREEMENT (AGR :P (PLURAL))
                        (AGR :P (SINGULAR))
                        (+CONJ :CONJTYPE))
                        → ∅
(SUBJECT-VERB-AGREEMENT (AGR :P (PLURAL))
                        (AGR :P (SINGULAR))
                        (-CONJ))
                        → ∅
(SUBJECT-VERB-AGREEMENT (AGR :P (SINGULAR))
                        (AGR :P (PLURAL))
                        (-CONJ))
                        → ∅
```

Figure 2.4: A Sample Constraint Relation 3: SUBJECT-VERB-AGREEMENT

The first of these solutions enforces standard subject verb agreement: whether the subject NP is a conjunction or not, the agreement features of the subject and the VP must be the same. The second allows the VP to bear the singular feature when the subject is plural, just in case the subject is a conjunction. The last two rules allow the subject and the VP to disagree, when the NP is not a conjunction. The statement about the conjunction status of the subject is necessary in these last two solutions to make them orthogonal to the first two, so that a single structure will not be unnecessarily analyzed with more than one solution.

This mechanism is superior to simply not requiring a VP to agree with its subject at all, by using distinct variables for the agreement features of the subject NP and the VP, since, given a large enough corpus, we can automatically associate different probabilities with the different solutions of this constraint relation. This is particularly useful in a spoken language system, since this will allow all the possibilities, with some degree of probability, but will always prefer the most common solution and will only choose a less likely solution if a more common one is unavailable.

2.2.1 Constraint Relation Compilation

An analysis of the time spent in parsing showed that substantial time was spent in processing constraint relations via standard unification, and it became clear that more efficient data structures and procedural techniques could be used to implement many such computations. Therefore, we have developed a mechanism, in addition to our standard rule formalism, that permits the creation of constraint relations that will be compiled into executable LISP code. From the standpoint of the rest of the grammar, these constraint relations look exactly like any other constituent: they return either a substitution list if the constraint succeeds or the designated symbol **FAIL** if it does not. However, the ability to define elements in the grammar that in fact are executed as code means that, in effect, we have a grammar consisting of declarative elements that can contain associated procedures. For example, while the P-MIN and NPTYPE-FILTER constraint relations were presented in their current, declarative form, we have found it more efficient to redefine the SUBJECT-VERB-AGREEMENT constraint relation in a procedural version.

The ability to define procedural elements as part of grammar rules is crucial for the recasting of the grammar using relational labels, discussed below in Section 2.4 and in Section 3.5. Without the ability to attach procedures to our rules that check for the coherence of local syntactic and/or semantic representation, our current system would have been impossible.

2.3 Subgrammar Development

In the earlier version of the grammar, optional elements were handled in the following manner. Special categories were introduced to simulate optionality. By convention, such categories had names that began with **OPT**. These nodes were used to implement optionality in the following way. First, there was always a rule of the form:

OPT<*category*>
$\rightarrow \emptyset$

that is, a rule expanding to nothing. The inclusion of such a rule was what allowed categories to be optional in the first place. In addition, there were always one or more rules of the form:

OPT<*category*>
$\rightarrow$
$<cat_1>$ ...

where $<cat_1>$ is a category of the grammar.

To allow multiple occurrences of the same optional category, such optional nodes were produced recursively:

```
OPT<category>
→
<cat1 >
OPT<category>
```

While we have retained a few such dummy categories, mostly for non-iterative optional categories, we have developed another scheme that is used quite generally throughout the grammar for elements that may be optional and freely ordered. This scheme is to make the categories that contain optional modifiers, complements, and adjuncts recursive, and to write a small number of rules, each one introducing a single optional constituent. This device immediately allows for optionality, iteration, and free order of elements. Moreover, it eliminates an unpleasant side-effect of our earlier treatment of optionality. Optional positive adjectival modifiers of NP were introduced via the category OPTADJP. Since various other elements, such as present participles ("flying") and passive participles ("booked") also functioned as pre-nominal modifiers, such elements were "lifted" to Adjectives or Adjective Phrases by chain (unit-production) rules, to allow them to function as nominal modifiers and also to give them the correct semantics. In our current version of the grammar, all prenominal modifiers are introduced by rules of the form:

```
(N-BAR ...)
→
<pre-nominal-modifier>
(N-BAR ...)
```

<pre-nominal-modifier> currently includes:

positive adjective phrases, as in "a *cheap* flight".

common nouns, as in "the *Boston* flight".

unit noun-phrases, as in "*two hundred dollar* one way tickets".

time expressions, as in "the *3 o'clock* flight".

date expressions, as in "the *November ninth* flights".

passive participles, as in "a *confirmed* reservation".

present participles, as in "a *departing* flight".

We have adopted a similar scheme for the post-head modifiers and complements of N. In the case of NP, we first created the category CORE-NP, which consists of all the left modifiers of an NP up to and including the head, ("the flights", "the evening flights", "the cheapest evening flights", etc.) Next, we added the chain rule:

```
(NP ...)
→
(CORE-NP ...)
```

Finally, we added recursive NP rules, each adding one of the types of constituents that can follow the head noun:

relative clause, as in "a flight *that leaves after 3 PM*".

adjective phrase, as in "a flight *non-stop* from Boston to San Francisco".

present participial phrase, as in "a flight *arriving before 3 PM*".

passive participial phrase, as in "a flight *delayed due to weather*".

a "path" expression, as in "a flight *Boston to San Francisco*".

One minor change to the NP grammar that we have introduced is the category UNIT-CORE-NP. This is a Noun Phrase whose head is a unit of measure (e.g. "foot", "dollar", "pound", "year", etc.) and which appears pre-nominally. In this position, the head noun remains singular, no matter the number of its specifier: compare "Show me the flight that costs *five hundred dollars*" with "Show me the *five hundred dollar* flight". This category is highly restricted in its distribution, being limited to the recursive N-BAR rule introduced above and an Adjective Phrase rule that specifies that the Adjective Phrase appears pre-nominally.

We have also adopted a similar scheme for the post-modifiers and complements of ADJP. CORE-ADJP is a much more impoverished category than CORE-NP, however, since the left modifiers of Adjectives are pretty much limited to a type of constituent we have labelled DEGREESPEC (words like "so", "very", "too", etc.) and unit NPs ("*five years* old") and UNIT-CORE-NPs ("*five year* old"). The right modifiers and complements are also more restricted than in NP.

Finally, our most spectacular use of this recursive, "layers of the onion" structure is in the case of subcategorization of the arguments to V in VP, described in full detail in Section 3.4. All together, these grammar changes combined with a shift of most rules introducing terminal elements, such as the prepositions, the determiners, etc., from the grammar to the lexicon has resulted in a reduction in the size of the grammar with no loss in coverage and, in fact, with an actual increase in coverage. The overall size of the grammar has gone from over 1100 rules to approximately 450. In the case of VP rules, we have gone from over 80 rules to 15.

2.4 The Use of Labelled Arguments

While the earlier version of DELPHI utilized semantic interpretation as a post-process on a parse produced by syntax alone, during the course of this project we have introduced semantic elements into the grammar and made semantic processing part of the parsing process. The full details of these changes are presented in Chapter 3. Here, we discuss just one of these changes, since it affects the form of grammar rules. In the current version of the DELPHI grammar, rules no longer consist merely of a left hand side, which contains a single grammatical constituent, deriving a right hand side, which is a set of one or more grammatical constituents. Rather, each element on the right hand side is preceded by a label, indicating its grammatical relation in the structure produced. Such grammatical labels include such familiar grammatical relations as "head", as in:

```
(N-BAR ...)
→
:HEAD
(N ...)
```

or "subject", as in:

```
(ROOT-S ...)
→
...
:SUBJECT
(NP ...)
:HEAD
(VP ...)
...
```

In structures which do not have a head in the traditional syntactic sense, such as conjunctions, a dummy head may be inserted, whose semantics is computed by the syntax on the basis of general rules. Thus, the NP conjunction rule introduced above in Figure 2.1, has the form shown in Figure 2.5 in the current grammar, where {*NP*} consumes no input, but is a compiled constraint relation that produces the semantics for the entire conjoined NP.

```
(NP (AGR :P (PLURAL)) (REALNP :REALZ) :POSS :WH ...:DET-CLASS ...)
→
:ARG1
(NP (AGR :PERSONX :NUMX) (REALNP :REALX) :POSS :WH ...:DET-CLASSX ...)
:CONJ-ELEMENT (CONJ ...)
:ARG2
(NP (AGR :PERSONY :NUMY) (REALNP :REALY) :POSS :WH ...:DET-CLASSY ...)
:HEAD
{*NP*}
{P-MIN :PERSONX :PERSONY :P}
{NPTYPE-FILTER :REALX :REALY :REALZ}
```

Figure 2.5: A Grammar Rule for Conjoined Noun Phrases

In our current approach, we view the parser not as a device for constructing syntactic trees, but as an information transducer that makes it possible to simplify and generalize the rules for semantic interpretation. The purpose of syntactic analysis in this view is to make information encoded in ordering and constituency as readily available as the information encoded in the lexical items, and to map syntactic paraphrases to informationally equivalent structures. The actual interface between parsing and semantics is a dynamic process structured as a cascade (as in Woods' notion of cascaded ATNs [98]), with parsing and semantic interpretation acting as coroutines. The input to the semantic interpreter is a sequence of messages, each requesting the "binding" of some constituent to a head. The semantic interpreter does not perform any sort of recursive tree-walk over the syntactic structure produced by the parser, and is in fact immune to many details of the tree structure; see Section 3.5.

The chief effort over the last year has been to codify this notion of binding or "logical attachment", simplifying the set of such attachments to highlight the common underlying substructure of grammatical paraphrases. To this end we re-oriented our grammar around the notion of "grammatical relation". These relations may be seen as the end result of making the information encoded in ordering and constituency explicitly available. (In languages with freer word order this information is often encoded in morphological affixes or pre and post positions.) From the point of view of a syntactic-semantic transducer, the key point of any grammatical relation is that it licenses (one of) a small number of semantic relations between the ("meanings" of) the related constituents. Sometimes the grammatical relation constrains the semantic relation in ways that cannot be predicted from the semantics of the constituents alone (given "John", "Mary" and "kissed", only the grammatical relations or prior world knowledge determine who gave and who received). Other times the grammatical relation simply licenses the one plausible semantic relation (given "John", "ate" and "hamburger", if there is a relation, it is the hamburger that is most likely to have been consumed—but in the sentence "John ate the fries but rejected the hamburger" our knowledge of the destiny of the hamburger is mediated by its lack of a grammatical relation to "ate").

With this brief introduction, we now turn to a more detailed view of the semantics of DELPHI.

Chapter 3

Semantic Interpretation

In this chapter we describe the developments in semantics during the course of the project. A major change from our earlier work is that interpretation in DELPHI has shifted from a Montague Grammar [65] style rule-for-rule approach to one that carries out semantic interpretation directly during the course of the parse, rather than as a post-process on a finished parse tree [24, 25]. Meaning representations are thereby constructed, and semantic filtering constraints applied, as part of parsing the utterance.

This move has several desirable attributes:

- More information is available as input to semantic interpretation, so it is possible to gain higher coverage.
- Syntax and semantics are integrated, so semantic filtering constraints can be applied as constituents are built and attached, which results in more efficient processing.
- This integration is simple and does not require any complex engineering of cooperating software modules.

All three are important for a spoken language system.

Our chief aim has been to make this integration of semantic interpretation with the parsing process as efficient, simple, and generalizable as possible. This goal has led us to restrict the use of unification to just those situations in which it is the most appropriate tool, and to employ constraint relations, especially compiled constraint relations, discussed above in Section 2.2.1, in many places. As a case study of the evolution of DELPHI's syntax-semantics interface from a fairly vanilla "unification semantics" implementation, to the current state of affairs, summarized in Section 3.5, the bulk of the rest of the chapter is devoted to the evolution of the treatment of verbal subcategorization phenomena during the course of the project. We conclude this discussion with an overview of our general use of labelled arguments, introduced above in Section 2.4, as an efficient means of interfacing syntax with semantics in areas other than subcategorization. The earlier sections of this discussion necessarily refer to aspects of DELPHI which are no longer current. Those who wish to omit the historical background and justification for our current approach can skip directly to Section 3.5.

We have also included a section (3.6) that discusses semantic problems which were treated in our earlier, unification-based approach to semantics. We have included this historical material, which describes aspects of the system which have been superseded, for two reasons. First, without the necessary background explaining the problems with the most straightforward strategies for integrating syntax and semantics, the reasons for our move to the novel architecture of Section 3.5 might not be apparent. Second, some of the solutions presented in the historical sections, such as the Q-TERM and NOM-SEM structures, which we have discarded, may nevertheless be interesting as representations for such semantic information. Moreover, many of the problems described in these historical sections must be solved in any system for semantics, and so are useful as a reference.

3.1 Virtual Rules and Subcategorization

As a prelude to our discussion of the treatment of subcategorization in DELPHI, we begin with an overview of some of the standard treatments of subcategorization. In complex-feature based grammar formalisms such as DELPHI's, there is no formally separate lexicon; lexical items are introduced by phrase structure rules just as syntactic categories ("non-terminals") are. For example, in order for the word "show" to appear in the version of DELPHI's grammar used at the start of this project, there would need to be a rule of the following sort:

```
(V (NO-CONTRACT) (DITRANSITIVE (TAKES-ACTIVE)) (AGR :P :N) (BASE))
→
(show)
```

This rule states that "show" is the base form (`(EDPARTICIPLE)`), unspecified for person and number agreement (`:P :N`), that it takes a ditransitive complement structure (`(DITRANSITIVE)`), and may appear only in active constructions (`(TAKES-ACTIVE)`). Note that this rule introduces "show" in only one of its uses, the ditransitive (as in "Show me the flights from Boston to Denver"). There need to be analogous rules for its other uses, as well, such as simple transitive (as in "Show the flights from Boston to Denver"), and Direct Object plus Prepostional Phrase (as in "Show the flights from Boston to Denver to me.")

For efficiency purposes, DELPHI does not store rules introducing lexical items, but rather generates them as needed by the parser on the basis of information stored in the lexicon and in conjunction with a morphology program that handles the regularly inflected forms[1]; such rules that are created on demand but not permanently stored are often referred to as "virtual rules". Thus, while the lexicon has no formal place in our system, it is used as a repository of lexical information (subcategorization, semantics, morphology, etc.) that is used to construct the virtual rules that the grammar actually uses.

3.1.1 Subcategorization

Virtual rules are also used to ensure that a member of a lexical category, such as V (Verb), appears with the correct *complements*. Complements are so called, in traditional grammar, because they "complete the meaning" of a lexical item in some way. For example, a transitive verb requires a noun phrase to follow it: "Show the flights" is grammatical but "*Show" (* indicates ungrammaticality) is not. Complements are lexically specified in that a given lexical item may or may not require (or permit) a particular category. Thus, non-transitive verbs forbid a following noun phrase but may optionally permit other complements; e.g. "*Delta flies Boston" is ungrammatical, but "Delta flies to Boston" is fine.

The set of complements that a lexical item requires is often referred to as a *subcategorization frame*. The version of the DELPHI grammar used at the beginning of this project contained a rule for every subcategorization frame in which a verb might appear. Each such rule is "indexed", as it were, by a mnemonically named feature that must appear as the value of the subcategorization feature of any verb that can occur in that frame. (In this, it followed GPSG [37], which uses a similar indexing scheme.) The two rules in Figure IndexSubcatFigure illustrate this.

The first rule states that a VP may consist of a V (verb) followed by no other complements if the V bears the feature **(INTRANSITIVE)**. The second rule says that a VP may consist of a V followed by an NP just in case the verb is specified as being **(TRANSITIVE)** and is in the active voice (**((TAKES-ACTIVE))**). Clearly, such a treatment of subcategorization has the following properties:

[1]Irregularly inflected forms are listed in the lexical entry of their base form.

```
(VP ...)
→
(V ... (INTRANSITIVE ...) ...)

(VP ...)
→
(V ... (TRANSITIVE ... (TAKES-ACTIVE)) ...)
(NP ...)
```

Figure 3.1: Examples of "Indexed" Subcategorization Rules

1. It requires the existence of a separate VP rule for every subcategorization frame.
2. In cases in which a verb realizes the same semantic argument with different syntactic forms (e.g. "show" can realize the entity to which an object is shown as either an NP indirect object or a "to" PP), a separate subcategorization frame for each separate realization must appear in its lexical entry.
3. In cases where a verb takes a particular complement optionally, the lexical entry for that verb must specify separate subcategorization frames for its presence and its absence. For example, in the examples above "show" needed to be specified as `TRANSITIVE` for its use without an indirect object and `DITRANSITIVE` for its use with an NP indirect object.
4. In cases where the order of complements is free (as in "fly from Boston to Denver", "fly to Denver from Boston"), we require either that individual lexical items specify the different orderings or that the grammar automatically provide multiple rules, when the freedom of ordering is predictable. For example, if it is the case that all verbs that select two prepositional phrases permit them to appear in either order, our grammar might then contain two versions of the rule introducing double PP complements, as a sort of "lexical redundancy rule":

```
(VP ...)
→
(V ... (DITRANSPREP :PREP :PREP1 ...) ...)
(PP :PREP ...)
(PP :PREP1 ...)

(VP ...)
→
(V ... (DITRANSPREP :PREP :PREP1 ...) ...)
(PP :PREP1 ...)
(PP :PREP ...)
```

The first property entails that the number of subcategorization frames will be large if the number of actually occurring subcategorizations is large. In the former version of the DELPHI grammar, there were on the order of 45 subcategorization frames. Since our grammar formalism does not include meta-rules (a mechanism for deriving related versions of rules), there were over 80 actual subcategorization rules, to accommodate alternative forms, such as passive.

Note moreover that this indexing approach to subcategorization requires that the grammar contain a separate rule even for subcategorization frames that are associated with a single lexical item. For example, in English, only the verb "bet" seems to take two NP complements followed by a clause (as in "We bet him $5 that he could not think of such a verb.") As a dual of this, if we encounter a subcategorization pattern we have not seen before, we must introduce a new subcategorization rule, otherwise we cannot handle it.

As the interactions between the optionality of a verb's arguments and their freedom of ordering becomes complex, the number of separate subcategorizations it is required to specify can become unwieldy and hard to maintain, since changes to what is essentially the same meaning must propagate to all the subcategorization frames.

Finally, this approach to subcategorization assumes a rigid distinction between complements (which are lexically specified) and adjunct modifiers (which are usually treated as free). For example, in general English "fly" is considered a verb of motion, which requires a destination complement in the form of a "to" PP, takes an optional source complement in the form of a "from" PP, and, like all active verbs, permits optional time modifiers such as "at 5 PM". However, in a travel domain such as ATIS, which involves scheduled travel times, such "free" adjuncts occur much more frequently than in standard English and, moreover, are given a domain particular interpretation (e.g. the scheduled departure time of a flight, not the mere accidental time at which that flight happened to be flying). Hence, there seems to be more structure and regularity to the interpretation of adjunct modifiers than the strict complement-adjunct distinction allows.

Because of these facts, we have moved to a labelled argument approach, which provides greater flexibility. The rest of this chapter uses the development of our treatment of subcategorization as an example of how we integrated semantic processing with syntactic processing in the most efficient manner.

3.2 A Unification Semantics Treatment of Subcategorization

Our first attempt at merging semantic interpretation with parsing was to carry out semantic interpretation directly in the unification grammar rules themselves. This was accomplished by adding semantic features to the grammar rules, placing them on the same footing as the existing syntactic features. First, consider the following grammar rule, an example of the "indexing" approach to subcategorization already discussed.

```
(VP (AGR :P :N) :MOOD (WH-) :TRX :TRY)
→
(V :CONTRACT (TRANSITIVE) :P :N :MOOD)
(NP :NSUBCATFRAME (WH-) :TRX :TRY)
```

This rule states that a VP may consist of a transitive verb (`(V ... (TRANSITIVE) ...)`) followed by an NP.

Next, we demonstrate how we added semantic features to this VP rule. These new features are underlined:

```
(VP (AGR :P :N) :MOOD (WH-) :TRX :TRY :SUBJ :WFF)
→
(V (TRANSITIVE :WFF :SUBJ :OBJ) :P :N :MOOD)
(NP :NSUBCATFRAME (WH-) :TRX :TRY :OBJ)
```

This rule passes up a formula as the semantics of the VP, indicated by the variable `:WFF`. The semantics of the subject of the clause, indicated by the variable `:SUBJ`, is passed down to the verb, as is the semantics of the object NP, indicated by the variable `:OBJ`.

For the transitive verb "show", we have the following lexical rule:

```
(V (TRANSITIVE (SHOW' :SUBJ :OBJ) :SUBJ :OBJ) :P :N :MOOD)
→
(show)
```

We can think of this rule in functional terms as taking semantic arguments `:SUBJ` and `:OBJ` and returning as a value the wff (well-formed formula) `(SHOW' :SUBJ :OBJ)`. Note the placement of semantic arguments to the verb inside the subcategorization term (headed by the functor `TRANSITIVE`) instead of at the top-level of `V`. This means that a verb with a differing number of arguments, such as "give", has a different subcategorization functor with a corresponding number of argument places for the semantic translations of these arguments.

3.2.1 The Use of Constraint Relations for PP Interpretation

The facility for case analysis provided by constraint relations for syntax (Section 2.2) is also used in semantic interpretation, where the meaning representation of constructions are computed in terms of values of certain features which cannot always be known in advance. As a good example of the utility of constraint relations for semantic interpretation, let us consider the interpretation of Prepositional Phrases. Prepositional Phrases in both post-copular and other positions within VP are very common in most domains, including ATIS. Some examples include:

Which flights are on Delta Airlines
Which flights are on Thursday
Which flights are after 4 pm

The grammar rule generating a post-copular PP in our first introduction of semantic features was:

```
(VP :SUBJ :WFF)
→
(V (BE))
(PP :PP)
{PREDICATIVE-PP :PP :SUBJ :WFF}
```

PREDICATIVE-PP is the constraint relation in the rule. It is responsible for specifying the formula meaning of the VP in terms of the translation of the PP (`:PP`) and the translation of the subject passed down from the clause (`:SUBJ`).

The PREDICATIVE-PP solution for the "flight-on-airline" sense is as follows:

```
(PREDICATIVE-PP (PP-SEM (:OR (ON)(ABOARD) (ONBOARD)) (:NP AIRLINE))
                (:SUBJ FLIGHT)
                (EQUAL (FLIGHT-AIRLINE-OF :SUBJ) :NP))
→ ∅
```

The first occurrences of the variables `:NP` and `:SUBJ` above are paired with semantic types AIRLINE and FLIGHT; this is shorthand for the actual state of affairs in which a term representing a package of information (including encoding of semantic type) appears in the slots of the rule these variables occur in:

```
(Q-TERM ⟨QUANTIFIER⟩ ⟨VARIABLE⟩
        (NOM ⟨PARAMETER⟩ ⟨SET⟩ ⟨SORT⟩))
```

This structure is so constructed as to not unify with another such structure if its semantic type is disjoint, using a method for encoding semantic types as terms described in Section 3.6.2. For full descriptions of the Q-TERM, NOM, and PP-SEM functors, which are not used in our current work, see Section 3.6.1.

The PP translation is also a package with the functor PP-SEM, containing the preposition and the translation of the NP object of the PP. No local attempt is made to translate the preposition.

When parsing with the above predicate PP rule, the system searches through a database of PREDICATIVE-PP solutions like the above, much as a PROLOG-based system would. If a solution successfully unifies, the formula is passed up as the translation of the VP.

Constraint relations are used not only to impose a stipulation on the constituents of a rule but also to allow for multiple ways to satisfy these constituents. For example, the PP "on American Airlines" can apply differently to different NPs: to an NP headed by "flight", in which case it indicates that the flight is an American Airlines flight, or to an NP headed by "fare", in which case it indicates that fare is the fare of an American Airlines flight.

So far we have not indicated how the system would distinguish between these two cases: in other words, how it would tell a fare and a flight apart. The variables `:SUBJ` and `:OBJ` in the previously presented lexical rule for "hire" are typed to range over the Q-TERM structures that represent noun phrase semantics, which we introduced earlier.

As an example, we give a second version of the rule for "show" in Figure 3.2, this time incorporating the selectional restriction that a person shows a flight, one of the possible transitive senses of "show". The use of the numbers "1" and "2" above is intended to indicate the multiple occurrences of the complex forms they label. (Note that this is simply the Common Lisp [94] convention for re-entrant list structure in the rule above. This is at present only used for notational compactness; the system does not currently attempt to take computational advantage of re-entrancy during unification or other processing.)

```
(V (TRANSITIVE
     (SHOW' #1=(Q-TERM :Q1 :VAR1 (NOM :PARS1 :SET1 (PERSON)))
            #2=(Q-TERM :Q2 :VAR2 (NOM :PARS2 :SET2 (INANIMATE (FLIGHT)))))
     #1#
     #2#
   :P :N :MOOD))
→
(show)
```

Figure 3.2: Unification Semantics for "show"

3.3 The Treatment of Optional Complements

While the introduction of semantic features into our standard "indexing" treatment of subcategorization provided a basic semantics, it did not solve any of the problems relating to optionality of complements, freedom of ordering, and alternative realizations, which we discussed in Section 3.1.1. We now turn to a further development of our treatment of subcategorization: the introduction of mechanisms to handle optional complements in a more efficient way than simply enumerating subcategorization features in individual lexical items. Recall our initial treatment of the semantics of subcategorization as exemplified in the following rule:

```
(VP :SUBJ :WFF)
→
(V (TRANSITIVE :WFF :SUBJ :OBJ))
(NP :OBJ)
```

The semantics of the subject of the sentence is passed down through the `:SUBJ` variable to the V, along with the semantics of the complements. The V in turn passed back up the formula representing the semantics of the whole sentence, through the `:WFF` variable.

As pointed out above in Section 3.1.1, this mechanism requires one VP rule for every subcategorization frame one wants to handle, and this can require many rules. A more serious problem, however, arises in the case of optional complements to verbs, as seen in the following actual ATIS training sentences that use the verb "arrive":

Show me all flights from Boston to Denver that arrive before 5 PM
Show me flights ... that arrive in Baltimore before noon

Show me all the nonstop flights arriving from Dallas to Boston by 10 PM ...
Show me flights departing Atlanta arriving San Francisco by 5 PM
Show me flights arriving into Atlanta by 10 PM from Dallas

We see here that, in addition to the temporal PP that always accompanies it, "arrive" can be followed by (1) nothing else, (2) a PP with "in", (3) a "from"-PP and a "to"-PP, (4) a bare noun phrase, or (5) an "into"-PP and a "from"-PP. The principle pattern that emerges is one of complete optionality and independence of order. Indeed, in the fifth example, the temporal PP, which might be more traditionally regarded as an adjunct rather than a complement, and thus as one of the siblings of the VP rather than one of its constituents, is instead interposed between two PPs complements, making the adjunct analysis rather problematic.[2]

The only way the subcategorization scheme presented above could deal with these variations would be to enumerate them all in separate rules. But this would clearly be infeasible. The initial solution to this problem that we adopted constructed a right-branching tree of verbal complements, where the particular constituents admitted to this tree are controlled by constraint relations keying off the lexical head of the verb. There were two main rules, which are shown in Figures 3.3 and 3.4. The rule in Figure 3.3 generates any number of optional PP complements to a verb that otherwise takes no other complements, while that in Figure 3.4 generates any number of optional PP complements to a verb that takes an initial NP complement.

```
(VP :SUBJ (AND :INITIAL-WFF :COMP-WFF))
→
(V :LEX (INTRANSOPTCOMPS :SUBJ :INITIAL-WFF))
(OPTCOMPS :LEX :SUBJ (DUMMY) :COMP-WFF)
```

Figure 3.3: Optional Intransitive PP Complements

```
(VP :SUBJ (AND :INIIIAL-WFF :COMP-WFF))
→
(V :LEX (TRANSITIVEOPTCOMPS :SUBJ :OBJ :INITIAL-WFF))
(NP :OBJ)
(OPTCOMPS :LEX :SUBJ :OBJ :COMP-WFF)
```

Figure 3.4: Optional Transitive PP Complements

[2]To see that these PPs are truly associated with the verb rather than somehow modifying the subject flight-NP, one need only replace the subject with the pronoun "it".

The category OPTCOMPS generates a right-branching tree of optional PP complements. Consistent with the use of such dummy categories to introduce optional elements, as discussed in Section 2.3, OPTCOMPS is expanded by two rules.

```
(OPTCOMPS :LEX :SUBJ :OBJ (AND :WFF1 :WFF2))
→
(PP :PP)
(OPTCOMPS :LEX :SUBJ :OBJ :WFF1)
{OPTCOMP-PP :LEX :SUBJ :OBJ :PP :WFF2}
```

introduces the optional PP complements, while

```
(OPTCOMPS :LEX :SUBJ :OBJ (TRUE)) → ∅
```

expands to nothing, allowing the optionality.

OPTCOMP-PP is the constraint relation; it keys off the lexical head of the verb (provided as the binding of the variable `:LEX`) and combines the subject, object, and PP complement translations to produce the contribution of the PP complement to the final formula that represents the sentence meaning. An arbitrary number of PP complements are provided for by the recursion of the first rule above, which bottoms out in the case of the second rule when there are no more complements. Phrasal types other than PPs are accommodated by similar rules.

The solution for PP complements to "arrive" such as "in Atlanta", "into Baltimore" "at Denver" etc. follows:

```
(OPTCOMP-PP (ARRIVE)
            (:SUBJ type FLIGHT)
            (:NP type CITY)
            (:OR (INTO) (IN) (AT))
            (EQUAL (DESTINATION-CITY :SUBJ) :NP))
→ ∅
```

This rule says that for a flight to "arrive" INTO, IN or AT a city means that the city equals the value of the flight's DESTINATION-CITY attribute. Semantic type information is here notated with a shorthand keyword `type`; in the actual system a partially-specified term that packages semantic type information in a specific slot was unified into such variables as `:SUBJ`, `:OBJ`, and `:NP`. Note also the use of disjunction (via `:OR`) to combine different prepositions together.

While the OPTCOMPS approach to complementation solved the problem of optionality and free order among complements, it did not provide a solution to the problem of alternate syntactic realizations of the same semantic argument, such as the realization of an indirect object either as an NP or a "to" PP, which was discussed in Section 3.1.1. In the next section, we present a new mechanism which solves all these problems.

3.4 The Mapping Unit Approach to Subcategorization

In this section, we introduce the "mapping unit" approach to representing subcategorization information. The advantage of this approach over the basic indexing treatment and the OPTCOMPS mechanism lies in its flexibility, a flexibility which in turn offers greater robustness of coverage with respect to unanticipated variations of a verbal argument pattern, and easier extension of coverage to new patterns. It handles in a quite natural way complement order variation, optionality of complements, alternative syntactic realizations of arguments, and metonymy. This is essentially the approach to representing subcategorization information which we currently employ, although the use of generalized relational labels discussed in Section 3.5 has changed the form of the actual rules used and the way in which semantic information is represented and combined.

3.4.1 Previous Approaches

Many past approaches have sought to represent subcategorization declaratively, often using an approach based on the unification of feature values. Such approaches as Definite Clause Grammar [74], Categorial Grammar [2], PATR-II [87], and lexicalized TAG [81] include in one form or another a notion of "subcategorization frame" that specifies a sequence of complement phrases and constraints on them. Some have also advocated using the feature system to encode semantic information (as for example [66]), and this, as was seen above, was our own initial approach to subcategorization.

"Mapping unit" subcategorization is partly inspired by these approaches, but it handles several kinds of variation in natural language utterances which cause difficulty for them. These forms of variation are not in any sense marginal phenomena, but are instead repeatedly seen in naturally derived data, such as that for the ATIS SLS common task domain. The phenomena fall into three classes:

The first is variation in argument order, as seen in

fly from Denver to Boston
fly to Boston from Denver

Such variation can be handled by the frame approach, but only at the cost of specifying one frame for each order. Besides such lexically-specific variation of order, other sources of order variation include interpolation of elements traditionally considered adjuncts ("What flights leave at 3 pm from Denver") and heaviness effects ("Show on the screen the fares and departure times of all the flights from Boston to Dallas").

The second is the optionality of arguments, and the different consequences thereof, including zero anaphora:

"What restrictions apply?"
(= apply TO SOMETHING IN CONTEXT)

default value:

"Show the flights."
(= show the flights TO ME)

existential quantification:

"fly to Boston" (AT WHATEVER TIME)

and independent truth-conditions:

"The Rockettes kicked."

Each verb that has optional arguments tends to have different preferences for what to do with the omitted argument places, as the above examples make clear. The frame approach can handle them, but again only at the cost of specifying multiple frames.

The third and final type of variation is the metonymic coercion of arguments, as seen below:

"What wide-body jets serve dinner?"
(= "What FLIGHTS on wide-body jets serve dinner?"
aircraft themselves do not "serve meals")
"What airlines fly to Dallas?"
(= "What airlines HAVE FLIGHTS to Dallas?"
airlines themselves don't "fly")

In both examples there is a superficial clash of types which is meant to be reconciled through the interposition of an implicit binary relation between the objects having those types. Our work postulates a distinction between two kinds of metonymy: "referential", where the argument is taken to be an indirect reference to an object of the proper type, and "predicative", where only the argument slot of the predicate is coerced and the referent is taken literally. This distinction will be discussed in more detail below.

Most verbs in the ATIS corpora ("fly", "arrive", etc.) have flight, source, destination, time of day, and day of the week arguments, most of which are not obligatory and can occur in almost any order. The number of frames necessary is combinatorially impractical, and to this situation the phenomenon of metonymic coercion, which makes the variation potentially open-ended, only provides the final blow. A fundamentally different framework from that of subcategorization frames is needed.

3.4.2 The "Mapping Unit" Information Structures

The central idea of the mapping unit approach is that there are several different types of subcategorization constraints, which ought to be represented as separate constraints, rather than enumerating the "Cartesian product" of their possible combinations in fixed patterns.

The basic building block is the "mapping unit", a structure which represents the constraints on a particular phrasal argument and the contribution this argument makes to the semantics of the clause. Mapping units do not "know" whether they are optional or not, or in what order they occur with respect to other mapping units; this information is instead represented in the grammar and in a larger structure called a "map", of which the mapping unit is a component.

Figure 3.5 gives an an example of a mapping unit. Each mapping unit has the four components shown here: a grammatical relation (SUBJECT, DIRECT-OBJECT, OTHER-PP etc), a syntactic pattern, a type requirement, and semantic role information. The syntactic pattern is a unification pattern, and thus retains all the advantages of being able to handle partial information that are associated with unification. The syntactic pattern (in the example just NP) also includes slots for semantic translation (`:trans`).

```
SUBJECT
(NP :trans)
(FLIGHT :trans)
(= FLIGHT-OF :trans)
```

Figure 3.5: An Example Mapping Unit

The semantic type requirement (here FLIGHT) is also enforced by unification, but separately, so that a failure due to semantic type clash can be distinguished from one that violates syntactic constraints. In this way, if the type requirement is not met, the mapping unit can be metonymically coerced, filling the semantic role not with the original complement translation but with an indefinite object related to it via a binary relation that resolves the clash. For example, the unit above could be coerced to accept an object of type AIRLINE via the binary relation FLIGHT-AIRLINE-OF, which maps flights to their airlines, thus handling the utterance "What airlines fly to Dallas".

A final separate representation is the contribution the semantics of the argument makes to the semantics of the clause, which is indicated by a semantic role constraint. The role is set (with the equality symbol =) in the case of ordinary complement arguments and restricted (with <) in the case of temporal or locative adjunct modifiers, which are not restricted to any pre-specified number of occurrences; e.g. "in Harvard Square at Out of Town News next to the foreign magazine section". A semantic role can only be set once in any given clause (which of course does not exclude it from being set to a conjunctive element), but can be restricted arbitrarily many times.

The mapping units are combined in a larger structure called a "map". Figure 3.6 presents a (much reduced) example for the verb "fly", in the ATIS domain. Every map has four components:

1. a labeled-argument predicate with typed roles
2. a collection of "mapping units"
3. a completion condition
4. a translation rule for the labeled-argument predicate

The labeled-argument predicate—in the example 'FLY1'—is the representation of the verb's "meaning", and has an assigned set of typed semantic roles which can appear in any application of the predicate (but which are not necessarily required to appear in every application).

```
((FLY1 FLIGHT-OF FLIGHT
       ORIG-CITY CITY
       DEST-CITY CITY)
 SUBJECT
 (NP :trans)
 (FLIGHT :trans)
 (= FLIGHT-OF :trans)

 OTHER-PP
 (PP (FROM) (NP :trans))
 (CITY :trans)
 (= ORIG-CITY :trans)

 OTHER-PP
 (PP (TO) (NP :trans))
 (CITY :trans)
 (= DEST-CITY :trans)
 completion (AND (FILLED FLIGHT-OF DEST-CITY)
                 (FILLED-OR-ANAPHOR ORIG-CITY))
 translation (p-and (flight-dest FLIGHT-OF
                                 DEST-CITY)
                    (flight-orig FLIGHT-OF
                                 ORIG-CITY)
                    (flight-departure-time
                            FLIGHT-OF
                            TIME-OF-DAY)))
```

Figure 3.6: A Example Map

The completion condition must be satisfied by any complete clause with the verb as head, and includes requirements on the instantiation of semantic roles.[3] In the example map, the FILLED completion predicate requires that the roles FLIGHT-OF and DEST-CITY be filled by a literal argument to the verb, while the FILLED-OR-ANAPHOR completion predicate allows the role ORIG-CITY to be be implicitly filled by a discourse entity. This condition allows "What flights fly to Denver from Boston?", "What flights fly from Boston to Denver" and "What flights fly to Denver?" but forbids "What flights fly?".

Other completion predicates include FILLED-OR-DEFAULT, which specifies a default value for a role, FILLED-OR-EXISTS, which generates an existential quantification over the range type if the role is unfilled, and GRAMMAR-REL-FILLED, which requires that a particular grammatical relation have been assigned. The unqualified optionality of a

[3]This requirement, in effect, implements the Functional Completeness Condition of LFG [28].

semantic role is indicated simply by leaving it out of the completion conditions.[4]

The fourth map component, the translation rule, converts labeled-argument predicate applications into ordinary logic expressions based on the roles they instantiate. Its separation from the rest of the map avoids duplicate specification of the details of logical form construction.

This last point is important because a map can have multiple units on the same semantic role to represent multiple syntactic realizations of it. For example, the utterance "fly DENVER to Boston" (meaning "fly FROM DENVER to Boston") can be accommodated simply by adding the following unit to the map above:

```
OTHER-NP
(NP :trans)
(CITY :trans)
(= ORIG-CITY :trans)
```

and the function of the translation rule is exactly the same.

Any semantic role can be filled only once, so that overgeneration from multiple mapping units assigning the same role is prevented.[5] For example, given the mapping unit for OTHER-NP just presented and the original map for FLY1 already presented, "fly Boston from Denver" would be ruled out since, given the existing mapping units, both "Boston" and "from Denver" would be attempting to fill the ORIG-CITY semantic role. Note that "fly Denver from Denver" is ruled out for exactly the same reason, even though, pre-theoretically, the "same" entity, Denver, is being used to fill the same semantic role twice. Note, then, that the constraint against multiple fillers for a single semantic role is a constraint on the mapping from syntactic elements to semantic roles, and not directly on that from semantic entities to semantic roles.

Certain grammatical relations can also be assigned only once in the derivation of any clause. These are the "major" grammatical relations —SUBJECT, DIRECT-OBJECT, INDIRECT-OBJECT. The OTHER-⟨*cat*⟩ relations can be assigned arbitrarily many times, subject only to the constraint that semantic roles be filled only once. Multiple mapping units that assign the same major grammatical relation are also allowed, subject only to

[4]Note that this assumption of optionality means that, as far as any given lexical item is concerned, any grammatical realization of a semantic role is optional, including SUBJECT. However, as is well known, English, unlike Italian or Greek, requires an overt subject in finite clauses. This requirement is imposed by the grammar, rather than by individual lexical items.

[5]This requirement, in effect, implements the Functional Consistency Condition of LFG [28]. This requirement and the requirement that completion predicates be satisfied, in effect, implements the Theta Criterion of GB theory [29].

the above constraint and of course to the constraint on the unique assignment of semantic roles. This is useful for handling certain types of polysemy—specifically, the semantic overloading of syntactic argument positions.

For modifiers which are normally treated as adjuncts, such as temporal or locative modifiers, our framework provides a notion of "free" mapping units associated with distinguished roles (such as TIME-OF-DAY, etc.). Such units do not have to be included in the map for individual lexical items whose labeled-argument predicate translation includes the role. In the example map above, TIME-OF-DAY is such a "free" mapping unit.

Finally, we should point out that while the map information structures can handle a considerable degree of variation, it is not necessary for any one map to handle all the possible variations associated with a verb. A verb can have multiple maps in the case of conventional lexical ambiguity, just as it can have multiple subcategorization frames in other approaches.

3.4.3 Use of Mapping Units in Grammar and Semantic Interpretation

Standard phrase structure rules augmented with features [74] are in our approach further augmented with the *non-constituent* predicates AVAILABLE, VP-BIND, and COMPLETE-WFF, along with the selector CONSTIT. The following is an example of the grammar rule that assigns DIRECT-OBJECT, reduced to include only features of interest:

```
(VP :MAP :BINDINGS2)
→
(VP :MAP :BINDINGS1)
{AVAILABLE DIRECT-OBJECT :MAP :BINDINGS1}
(NP :TRANS)
{VP-BIND DIRECT-OBJECT :MAP {CONSTIT (NP) (1)} :BINDINGS1 :BINDINGS2}
```

The predicate AVAILABLE takes a grammatical relation, a map, and a bindings list, which is a list pairing mapping units with role fillers. It is satisfied if there is a unit in the map with that grammatical relation such that the semantic role of that unit is not set in the bindings list, and the grammatical relation is not assigned in the bindings list (if it is a "major" grammatical relation).

The predicate VP-BIND takes a grammatical relation, a map, an entire constituent (retrieved by the function CONSTIT as seen above) an input bindings list and an output bindings list. If it succeeds, it produces a new bindings list containing an additional pair of unit and filler. It will succeed if it finds a free unit in the map that can be matched with the passed-in constituent both syntactically and semantically. VP-BIND will have more than one solution if it finds multiple units with these properties, in which case there are multiple parses. (Note that the AVAILABLE predicate is really only necessary to prevent the parser from looking for a constituent that would only wind up not being attachable to the VP.)

The pair of map and bindings effectively constitutes the meaning of the VP, and can be likened to a an application of lambda-expression (the map) to arguments (the bindings). The difference is that while the arguments to a regular lambda-expression can either be bound all at once or in some fixed order (e.g. through currying) the arguments to a map are referred to by label, and can be applied in any order we please.

Currently, the maps only provide optionality information, while the relative order of complements is enforced by the grammar via grammatical relations. This has the advantage that certain ordering constraints need only be stated once, as opposed to over and over again in map entries. An example of a rule imposing ordering constraints is the ditransitive VP rule, which handles "Show me the flights":

```
(VP :MAP :BINDINGS2)
→
(VP :MAP :BINDINGS)
{AVAILABLE INDIRECT-OBJECT :MAP :BINDINGS}
{AVAILABLE DIRECT-OBJECT :MAP :BINDINGS}
(NP :TRANS1)
{VP-BIND INDIRECT-OBJECT :MAP {CONSTIT (NP)(1)} :BINDINGS :BINDINGS1}
(NP :TRANS2)
{VP-BIND DIRECT-OBJECT :MAP {CONSTIT (NP)(2)} :BINDINGS1 :BINDINGS2}
```

The constraint that the subject precedes post-verbal complements is expressed by the clause-level S rule, which assigns the relation SUBJECT:

```
(ROOT-S (QUESTION) :MAP :BINDINGS2)
→
(NP :TRANS)
(VP :MAP :BINDINGS1)
{VP-BIND SUBJECT :MAP {CONSTIT (NP)(1)} :BINDINGS1 :BINDINGS2}
```

The completion conditions for the clause are enforced by the rule for the top-most node, START. This rule contains a predicate COMPLETE-WFF that takes a map, bindings list, and delivers an output formula:

```
(START (QUERY :WFF))
→
(ROOT-S (QUESTION) :MAP :BINDINGS)
{COMPLETE-WFF :MAP :BINDINGS :WFF}
```

COMPLETE-WFF enforces the completion conditions of the map and reduce's the map and bindings combination to a formula if these conditions are satisfied.

The formula to be generated is specified by the translation rule component of the map. This translation rule can really be regarded as a kind of meaning postulate for the predicate that is associated with it directly. It consists of an ordinary logic expression containing references to the argument labels of the predicate. Repeated below is the translation for the predicate FLY1:

```
(P-AND (flight-dest FLIGHT-OF DEST-CITY)
       (flight-orig FLIGHT-OF ORIG-CITY)
       (flight-departure-time FLIGHT-OF
                              TIME-OF-DAY))
```

To generate the formula, the fillers of the argument roles are substituted for these references. The P-AND is a meta-conjunction operator with the property that if any of the role references of one of its conjuncts are unfilled, that conjunct is left out of the final formula. In this way we are not required to generate an existential quantification for a missing argument place (as for example the departure time of the flight).

There are certain instances in which an existential quantification is generated, however. If a semantic role has merely been restricted instead of filled, a narrow-scope existential quantification is generated and the variable of this quantification substituted for the role reference in the translation rule. Thus for "Flight 1 flies before 3 pm" we would have:

```
(exists t (precede t (time 3 0 pm))
          (flight-departure-time (flight-no 1) t))
```

3.4.4 Predicative Metonymy

Another case in which narrow-scope existentials are generated is the case of predicative metonymy, in which the semantic role in question has been type-coerced to accept an argument of a type different from its restriction. In this type of metonymy, the referent of this argument does not change. Instead a relation is established between it and an indefinite, existentially quantified object of the proper type. Thus for "Delta Airlines flies from Boston to Baltimore" we would have:

```
(exists x flight (airline-of x Delta)
                 (and (orig-city x boston)
                      (dest-city x baltimore)))
```

The distinction between referential and predicative metonymy only becomes visible when the actual referents of NPs are sought, as in:

What airlines fly to Boston?
What wide-body jets serve dinner?

In the first, it is implausible that "airlines" is being used to refer to some set of flights, since every flight is on some airline and there is no constraint. In the second, "wide-body jets" is far more likely to refer to some set of flights, since not every flight is on a wide-body jet.

Predicative metonymy is an essentially local phenomenon, while referential metonymy is an essentially global one. Our present implementation assumes predicative metonymy only and allows only a limited set of binary relations. Processing is such as to prefer attachments that do not require metonymy, by assigning a lower probability at parse-time to parses which do require it; see Section 4.2 for discussion of the use of statistical mechanisms to control parse preference. This is necessary to exclude an unreasonable parse of:

Show flights to Denver on wide-body jets serving dinner.

That is, one in which the participial modifier "serving dinner" is attached to the Noun Phrase "wide-body jets", rather than to the containing Noun Phrase "flights to Denver on wide-body jets". The lower attachment (to "wide-body jets") would entail coercing "wide-body jets" to refer to flights, giving us a semantic interpretation involving reference to "flights on flights". The higher attachment (to "flights") entails no such coercion and no such interpretive problems.

3.4.5 Other Benefits

The combination of labeled-argument predicate and translation rule offers several benefits not yet mentioned. One is that a given predicate can be shared between different lexical entries which provide different syntactic realizations of it. For example, in the ATIS domain the verbs "depart" and "originate" have very similar core meanings, yet have semantic roles realized by different prepositions:

The flight departs from Boston.
The flight originates in Boston.
*The flight departs in Boston.

The words are not synonyms in the normal sense that one can be substituted for the other in such a way as to preserve grammaticality. But their common semantic content can be represented.

Another advantage is that a denotational semantics with optionality is implemented without requiring Davidsonian-style event quantifications [34]. While event objects make sense in some contexts, having an existential quantification over events for every verb is frequently inconvenient in further processing. Certainly it is so in the ATIS domain, where the chief semantic outcome of clauses seems to be a set of predications on attributes of flight-individuals and there really are no "events" as such at all. Event quantifications are not precluded, however—they could be produced with a different translation rule schema.

3.4.6 Comparison with Other Work

Our treatment of the syntactic aspects of subcategorization is most like that of PATR-II [87]. Both PATR-II and the mapping units approach use recursive VPs, with each level of VP structure, in effect, "peeling off" a single constituent of the head verb's complement list.[6]

[6]Our approach allows more than a single complement constituent to appear at a given level, however, as the ditransitive VP rule above shows.

A major difference between the two approaches is that the PATR-II system of subcategorization is essentially limited to popping constituents off the subcategorization list, in fixed order, requiring a separate subcategorization list for each variation in order. Our approach allows complements to be found in whatever order the grammar will allow them. In this respect, it is more like the UD system [50], which has an operator for "non-deterministic extraction from arbitrary list positions".[7] However, our system does not literally remove units from the map, but rather simply marks them as no longer available for binding. Moreover, the mechanism of allowing multiple syntactic realizations for a given semantic relation in a single representation of complement structure is, to our knowledge, unique.

Related work on the semantic aspects of argument optionality has been reported by Palmer [71], [33]. Our work differs from this mainly in the tighter coupling of syntax and semantics during processing and the use of recursive VP structures, which potentially allows for an elegant solution of cases of non-constituent conjunction, not addressed in the other work. Our system also has a finer grained treatment of optionality; whereas [71] and [33] divide argument roles into OBLIGATORY, ESSENTIAL, and NON-ESSENTIAL roles, we provide a richer set of possibilities. The decoupling of syntactic realizations of a role in a mapping unit from the semantic typing constraints on that role, to allow for different types of metonymic extension is also a distinction. On the other hand, the use of named thematic or semantic roles which cut across particular predicates in [71] and [33] might provide a more compact form for capturing linguistic generalizations, to the extent that such a theory of thematic relations is well-motivated.

3.5 Generalized Grammatical Relations

The use of labelled arguments in the grammar, introduced above in Section 2.4, is a development of and generalization of our use of mapping units for subcategorization. Note that, in a rule such as:

```
(VP :MAP :BINDINGS2)
→
(VP :MAP :BINDINGS1)
{AVAILABLE DIRECT-OBJECT :MAP :BINDINGS1}
(NP :TRANS)
{VP-BIND DIRECT-OBJECT :MAP {CONSTIT (NP) (1)} :BINDINGS1 :BINDINGS2}
```

the NP daughter of VP is clearly meant to be interpreted as the DIRECT-OBJECT. How-

[7]Other systems which have a similar mechanism include the Lilog system of IBM Germany and the MiMo machine translation system.

ever, that fact is not directly stated in the rule but is rather implied by the fact that this rule is only applicable in there is an available DIRECT-OBJECT and that building this rule binds this previously available DIRECT-OBJECT relation. Introducing relational labels directly into grammar rules (1) makes the grammatical relations of all the constituents of the rules obvious; (2) makes the former "available" and "(vp-)bind" relations implicit in the grammatical label (i.e. we cannot bind a grammatical relation unless it is available to be bound), a more natural state of affairs; and (3) allows us to develop a system of semantic interpretation which applies generally across various grammatical relations. We expand on these points in the rest of this section.

Recall that grammatical relations are incorporated into the grammar by giving each element of the right hand side of a grammar rule a grammatical relation as a label. Some typical rules are, in schematic form:

```
(NP ...)
→
:HEAD (NP ...)
:PP-COMP (PP :PREP ...)


(N-BAR ...)
→
:PRE-NOM (N ...)
:HEAD (N-BAR ...)
```

The first indicates that an NP may have an NP as a head and a PP as an adjunct, with the grammatical relation `:PP-COMP` holding between them (the actual operation of binding a `:PP-COMP` splits it into a number of sub-relations based on the preposition in a manner already seen in the discussion of constraint relations in Section 3.2.1, so that this face can be safely ignored here). The second indicates that the head need not occur as the first constituent on the right side. All that is required is that one of the right-hand elements is labeled as the "head" of the rule, and it is the source of information about the initial semantic and syntactic "binding state", the equivalent of the BINDING variable in our earlier mapping units scheme. This binding state controls whether or not the other elements of the right-hand side can "bind" to the head via the relation that labels them. Semantics is associated with grammatical relations, not with particular grammar rules (as are Montague-grammar and most unification-based semantics systems, including earlier versions of DELPHI's semantics).

3.5.1 "Binding rules"—the Semantics of Grammatical Relations

The implementation of the use of such binding states in the transduction of grammatical relations to semantic structure is facilitated by the compiled constraint relations we have introduced into our unification grammar formalism; see Section 2.2.1. A separate system of "binding rules" for each grammatical relation licenses the binding of a constituent to a head via that relation by specifying the semantic implications of binding. These rules generally specify aspects that must be true of the semantic structure of the head and bound constituent in order for the binding to take place, and may also specify certain syntactic requirements. They may take into account the existence of previous bindings to the head, allowing certain semantic roles (such as time specification) to be filled multiply, while other semantic roles may be restricted to having just one filler. Thus, given the use of generalized grammatical relations throughout the grammar, the rule introducing a direct object, which was presented above as:

```
(VP :MAP :BINDINGS2)
→
(VP :MAP :BINDINGS1)
{AVAILABLE DIRECT-OBJECT :MAP :BINDINGS1}
(NP :TRANS)
{VP-BIND DIRECT-OBJECT :MAP {CONSTIT (NP) (1)} :BINDINGS1 :BINDINGS2}
```

now has the much simpler form:

```
(VP ...)
→
:HEAD (VP ...)
:DIRECT-OBJECT (NP ...)
```

In effect, we may consider the relational label `:DIRECT-OBJECT` to be, in effect, a macro, which performs the functions of the previous constraint relations AVAILABLE, which made sure that the map of the head V contained an unbound Direct Object, and VP-BIND, which bound the direct object NP found to the Direct Object position in the head V's map, performing any necessary semantic type checking.

Note also that the semantic variables `:MAP`, `:BINDINGS1`, `:BINDINGS2`, and `:TRANS` do not appear in the second version of the rule. This is because our current implementation of the syntax-semantics interface no longer includes semantic variables as part of the term structure of syntactic categories. Rather syntactic rules and semantic information are bundled together in a record data structure (the Common LISP defstruct). Constraint relations that perform semantic operations, such as checking for the availability of a grammatical relation to bind, binding semantic roles, etc. are able to access this semantic information and perform any necessary operations (such as semantic type consistency checking). However, semantic interpretation in the current version of DELPHI makes no real use of the standard unification operation used to build syntactic structures.

As constituents are added to the recursive structure the binding list is extended. As layers are added to the "onion", a simple linear list of bindings is maintained representing the head and its grammatical relation to each of the constituents added with each layer of the onion. Semantic binding rules are used to verify the local semantic plausibility of a structure, i.e. the semantic plausibility of each proposed grammatical relation. The next phase of semantic interpretation takes place when the onion is complete, i.e. when a constituent X is inserted as other than the head of a larger constituent. This situation provides evidence that the outermost layer of the onion has been reached, and that no more adjuncts are to be added. At this time it is possible to evaluate semantic rules that check for completeness and produce an "interpretation" of the constituent. These completion rules operate directly on the binding list, not on the recursive left or right branching tree structure produced by direct application of the grammar. The actual tree structure is at this level immaterial, having been replaced by the flattened binding list representation of relational structure.

3.5.2 Robustness Based on Statistics and Semantics

Simply having a transduction system with semantics based on grammatical relations does not deal with the issue of robustness—the ability to make sense of an input even if it cannot be assigned a well-formed syntactic tree. The difficulty with standard syntactic techniques is that local syntactic evidence is not enough to accurately determine grammatical relations. An NP (e.g. "John") followed by a verb (e.g. "flew") may be the subject of that verb (e.g. "John flew to Boston") or may be unrelated (e.g. "The man I introduced to John flew to Boston"). The standard way of getting around this is to attempt to find a globally consistent set of grammatical relation labels (i.e. a global parse) and make use of the fact that the existence of a global parse containing a given relation is stronger evidence for that relation than local structure (although syntactic ambiguity makes even such global structures suspect). This is indeed the best approach if all you have available is a syntactic grammar.

The strategy we have developed in DELPHI is based on the existence of two other sources of information. In the first place we have semantic constraints that can be applied incrementally, so that we can check each proposed grammatical relation for semantic coherence in the context of other assumed grammatical structures. Additionally, we have statistical information on the likelihood of various word senses, grammatical rules, and grammatical-semantic transductions, as discussed in Section 4.2, below. Thus we can not only rule out many locally possible grammatical relations on the basis of semantic incoherence, we can rank alternative local structures on the basis of empirically measured statistics. The net result is that even in the absence of a single global parse, we can be reasonably sure of the local grammatical relations and semantic content of various fragments. We can even give numerical estimates of the likelihood of each such structure.

3.5.3 Advantages of This Approach

The separation of syntactic grammar rules from semantic binding and completion rules has important consequences for processing. First, it enables the notion of grammatical relation to be separated from the notion of tree structure, and thus greatly facilitates fragment parsing; see Section 4.3. Second, while it allows syntax and semantics to be strongly coupled in terms of processing (parsing and semantic interpretation) it allows them to be essentially decoupled in terms of notation. This makes the grammar and the semantics considerably easier to modify and maintain.

We believe, however, that in the long term the most important advantage is that this view leads us to a new kind of language model, in which knowledge can be much more easily extracted through automatic training. We view the role of the grammar as codifying the way that tree structure provides evidence for grammatical relations. Thus the rule

```
(NP ...)
→
:HEAD (NP ...)
:PP-COMP (PP :PREP ...)
```

says that a noun phrase followed by a prepositional phrase provides evidence for the relation PP-COMP between the PP and NP head.

The separation between rule types will allow us for the first time to consider the effect of grammatical relations on meaning, independently of the way that evidence for these relations is produced by the parser. One effect of this is to make it possible to use a hypothesized semantic interpretation of a set of tree fragments to generate a new syntactic rule.

Thus, in normal operation, the primary evidence for a grammatical relation is the result of actually parsing part of an input. However, since grammatical relations between constituents entail semantic relations, if we can make an estimate of the likelihood of certain semantic relations based on domain knowledge, pragmatics, and task models, etc., it is in principle possible to use abductive reasoning to suggest likely grammatical relations, and thereby propose new grammar rules. In effect, grammatical relations form an abstract level of representation that greatly simplifies the interaction of syntactic and semantic processing.

3.6 Unification Semantic Interpretation in DELPHI

In this section, we detail various problems solved in the earlier, unification semantics, version of DELPHI. We believe that some of the mechanisms we implemented may be use to other researchers. Moreover, all of the semantic issues raised here must be faced in any treatment of semantics.

We first introduced semantic interpretation into grammatical analysis in DELPHI by adding semantic features directly into the grammar rules, as is common in complex feature based grammar formalisms. We describe here our initial efforts along these lines, some of which pre-dated the restructuring of the grammar described in Section 2.3. The facility for case analysis with constraint relations described in Section 2.2 is also used in semantic interpretation, where the meaning representation of constructions are computed in terms of values of certain features which cannot always be known in advance.

3.6.1 Analysis of Noun Phrases and Noun Modifiers

Figure 3.7 presents a simplified form of the rule for regular count Noun Phrases used in the version of our grammar to which semantic features were first added. Semantic features are underlined.

```
(NP :NSUBCATFRAME (AGR :P :N) :WH (Q-TERM :Q :VAR :NOM5))
→
(DETERMINER :N :WH :NOM1 :NOM2 :Q)
(OPTNONPOSADJP (AGR :P :N) :NOM4 :NOM5)
(OPTADJP (AGR :P :N) (PRENOMADJ) :NOM3 :NOM4)
(N-BAR :NSUBCATFRAME (AGR :P :N) :NOM1)
(OPTNPADJUNCT (AGR :P :N) :NOM2 :NOM3)
```

Figure 3.7: Unification Semantics for NP

This rule generates NPs that have at least a determiner and a head noun, and which have zero or more prenominal superlative or comparative adjectives ("cheapest", "less expensive" etc.), prenominal positive adjectives ("overnight", "alleged"), and adjuncts ("in Boston", "that arrive from Denver"). Its effect is to take **:NOM1**, the NOM-SEM semantics of the head noun passed up through N-BAR, and thread it through the various modifications, add a quantifier and a variable for quantification, and deliver the resulting package as the semantics for the whole NP.

NOM-SEM terms have the following structure. The principal functor of this type is NOM, which has the argument structure:

```
(NOM PARAM-LIST SET-EXP SORT)
```

The PARAM-LIST is a (possibly empty) list of parameters, used to indicate the free argument places in a relational noun. SET-EXP is a logical expression which denotes a set of individuals. SORT is a term structure which represents the semantic class of the elements of SET-EXP.

The initial NOM-SEM comes from the head N-BAR, and is indicated by **:NOM1**, the variable in that argument position. It is first of all passed to the DETERMINER. Along with a quantifier, **:Q**, the DETERMINER passes back a possibly modified NOM-SEM, **:NOM2**. The reason for this is that the determiner may be possessive, and a possessive determiner effectively functions as a noun modifier which enters into scope relations with other modifiers of the NP. Consider the noun phrase "John's best book". This cannot be analyzed as

```
(SET X (BEST' BOOK') (EQUAL (AUTHOR-OF X) JOHN'))
```

that is, as the subset of the best books in the world that also happen to be written by John. Instead, it must be analyzed as:

```
(BEST' (SET X BOOK' (EQUAL (AUTHOR-OF X) JOHN')))
```

that is, the best of the books written by John.

The essential point is that the possessive DETERMINER must carry out its modification before other elements of the NP can, yet must still follow all other modifications in affixing a quantifier to the final result of the NP. If the determiner is conceived of as just a higher-order function returning a single value, as in Montague Grammar, it is difficult to see how this can be done. The virtue of this approach is that it allows the determiner to return as separate values both a quantifier and a suitably modified nominal.

If the determiner is not possessive it simply passes up the same NOM-SEM it was originally given. The NOM-SEM returned by the DETERMINER, whether modified or not, is then passed down to the adjuncts of the NP as `:NOM2`, which modify it and return `:NOM3`. This is then passed to the regular (non-superlative) prenominal adjectives for further modification, returning `:NOM4`. Finally, `:NOM4` is passed to the constituent OPTNONPOSADJP, the optional superlative adjectives. The final NOM-SEM, `:NOM5`, is passed up to become an element of the complete Q-TERM semantics of the NP.

Q-TERMs have the following form:

```
(Q-TERM QUANTIFIER VAR NOM-SEM)
```

The QUANTIFIER is one of the many quantifiers that correspond to determiners in English: ALL, SOME, THE and various WH determiners. Proper NPs were treated as definite descriptions in this version of our system; they were thus represented using the THE quantifier.

The VAR denotes a variable of the object language, and is left uninstantiated (being filled in by a unique object-language variable by a quantifier module). The NOM-SEM represents the set that the quantification ranges over; it effectively represents the semantics of the head of the NP after modification by the NP's other constituents.

Note that the structure of Q-TERMs and NOM-SEMs means that the SORT field of the NOM is accessible, via one level of indirection, from the Q-TERM NP representation. It is this feature which provides the means for selectional restriction based on semantic class elsewhere in the grammar.

Semantic classes (arranged in a hierarchy) are represented as complex terms, whose arguments may themselves be complex terms. A translation (described in Section 3.6.2) is established between semantic classes and these terms such that non-empty overlap between two classes corresponds to unifiability of the corresponding terms, and disjointness between classes corresponds to non-unifiability of the corresponding terms.

As an example of the action of the modifying elements in the above rule, consider the following rule for generating an NP adjunct from a PP:

```
(OPTNPADJUNCT :NOM1 :NOM2)
→
(PP :PP)
{MODIFYING-PP :PP :NOM1 :NOM2}
```

The NOM-SEM passed in from the containing NP, `:NOM1`, is in turn passed down to a constraint relation, MODIFYING-PP, which takes the semantics of the PP, `:PP`, and "computes" the modified NOM-SEM, `:NOM2`, which is then passed back to the NP as the result of the modification. Note that PPs in our system are given only partial semantic interpretations, which consist of the preposition of the PP and the translation of the PP's NP object. Their representations are thus of the following form:

```
(PP-SEM PREP NP-SEMANTICS)
```

MODIFYING-PP is used to encompass different kinds of PP modification. Relational modification, where the PP essentially fills in an argument, is handled by the following solution to MODIFYING-PP:

```
(MODIFYING-PP (PP-SEM (OFPREP) :NP)
              (NOM (PARAM :NP) :SET :SORT)
              (NOM (NO-PARAM) :SET :SORT))
→ ∅
```

Since this rule is a constraint relation solution, its right-hand side is empty. It unifies the NP object of the PP with the "parameter NP" of the argument nominal. Of course, it will not be unifiable if the argument nominal does not contain a parameter NP, or if the parameter NP of the argument nominal contains the wrong semantic type.

The lexical rule introducing the relational noun "cost" in its application to ground transportation is as follows:

```
(N (NOM (PARAM #1=(Q-TERM :Q :VAR
                          (NOM :PARS :SET (GROUND-TRANSPORTATION-SUMMARY))
        (SETOF (COST' #1#))
        (INANIMATE (DOLLAR-AMT)))))
→
(cost)
```

Note the requirement that the filler of the slot be of sort GROUND-TRANSPORTATION-SUMMARY, and the co-occurrence of this filler inside the NOM's set expression.

Of course PPs can also occur in a predicative sense. For example, an airport can be "in" in a city. To handle this we have the following solution to to the constraint relation PREDICATIVE-PP:

```
(PREDICATIVE-PP
    (PP-SEM (INPREP)
            #1=(Q-TERM :Q1 :VAR1
                   (NOM :PARS1 :SET1 (INANIMATE (CITY)))))
    #2=(Q-TERM :Q2 :VAR2
                   (NOM :PARS2 :SET2 (INANIMATE (AIRPORT))))
    (EQUAL (CITY-OF #2#) #1#))
→ ∅
```

Note that this constraint solution will only unify if the class of the NP object of the PP unifies with CITY, and the class of the NP being predicated of unifies with AIRPORT.

When such a PP occurs as an adjunct to an NP, the derivation passes through the following indirect "lifting" rule:

```
(MODIFYING-PP :PP
    (NOM :PAR :SET :SORT)
    (NOM :PAR (SET :VAR :SET :WFF) :SORT))
→
(PREDICATIVE-PP :PP
    (Q-TERM (BOUND-Q) :VAR
       (NOM :PAR :SET :SORT))
    :WFF)
```

Although the right-hand side of the rule is in this case not empty, it will like all constraint relations derive the empty string in the end.

Similar distinctions of modificational power are seen in the case of adjectives, where an adjective like "average" or "previous" has the power to abstract over free parameters of the noun meaning, while an adjective like "female" does not. Consider the rule below:

```
(OPTADJP (AGR :P :N) :POSIT :NOM1 :NOM3)
→
(ADJP (AGR :P :N) :POSIT :ADJ-SEM)
(OPTADJP (AGR :P :N) :POSIT :NOM1 :NOM2)
{MODIFIYING-ADJ-READING :ADJ-SEM :NOM2 :NOM3}
```

This rule generates a string of one or more adjectives. Nominal semantics is threaded through the adjectives right to left. Adjective phrase semantic representations (ADJ-SEMs) come in two varieties:

```
(MODIFYING-ADJ NOM-SEM NOM-SEM)
```

and

```
(PREDICATIVE-ADJ NP-SEMANTICS WFF)
```

These represent different semantic types of adjective. Adjectives like "previous", with the power to modify the whole noun, have a semantic representation headed by the functor MODIFYING-ADJ, while adjectives like "female", which only operate upon individual elements of the noun's extension, have a representation headed by the functor PREDICATIVE-ADJ. The constraint relation MODIFYING-ADJ-READING accepts the first kind of adjective unchanged and lifts the second kind to the appropriate level. Note that while predicative PPs and adjectives can be "lifted" to the noun modifying level, the converse is not true. That is, the system does not allow "That value is previous" or "That salary is of Clark".

3.6.2 Other Aspects of DELPHI Unification Semantics

Encoding Semantic Classes as Terms

The translation from semantic classes to complex terms can be performed systematically. In this section we present an algorithm for translating semantic classes to terms, designed to work on taxonomies of semantic classes represented in a system such as KL-ONE [82] or NIKL [67]. It has the advantage, important from the point of view of such systems, that it correctly handles the distinction between "primitive" and "defined" classes — "defined" meaning that the class is simply an intersection of two or more other classes.

The algorithm is seen in Figure 3.8, where the main work is done by the function TRANSLATE.

Throughout, the symbol **:ANY** indicates a "don't care" variable, unifying with anything. This is in fact the only use of variables made. The operation REGULARIZE is used to remove non-primitive classes from the taxonomy, and set them aside. It is simple and we do not give it here.

We now consider the classes PERSON, MALE, FEMALE, ADULT, CHILD, MAN and PRIEST. MALE and FEMALE are disjoint sub-classes of PERSON, as are ADULT and CHILD. MAN is the class which is the intersection of ADULT and MALE. PRIEST is a sub-class of MAN, but not identical to it. Following are the translations the algorithm in Figure 3.8 gives to several of these classes:

```
PERSON → (PERSON :ANY :ANY)
ADULT → (PERSON (ADULT :ANY) :ANY)
MALE → (PERSON :ANY (MALE :ANY))
MAN → (PERSON (ADULT :ANY) (MALE :ANY))
PRIEST → (PERSON (ADULT (PRIEST))
                    (MALE (PRIEST)))
```

Essentially, the algorithm works by mapping each set of mutually disjoint children of the class to an argument place of the term to be associated with that class. The term associated with a class has the same depth as the depth of the class in the taxonomy.

The translation produces by this algorithm are similar to those produced by the algorithm by Mellish [63]. We claim two advantages for ours. First, and as already pointed out, it takes into account the difference between "if" (primitive) and if-and-only-if (non-primitive) axiomatizations, where it would seem that the Mellish algorithm does not. Second, it is simpler, not requiring such notions as "paths" and extensions "to" and "beyond" them.

As a final comment on the issue of encoding semantic classes as terms, we note that there is another encoding method which may have been overlooked: that is, encoding each class as a term which has the same number of arguments as there are classes. It works as follows. In the argument position corresponding to the class being translated put a "1", and put a "1" in argument positions corresponding to subsuming classes as well. In argument positions corresponding to disjoint classes put a "0". In all other positions put a "dont-care" variable. While perhaps using space inefficiently, this encoding will have all the desired properties.

Using Unification to Encode Semantic Constraints beyond Semantic Type

Not all constraints on meaningfulness are strictly reflections of the semantic type of phrase denotations. Consider the lexical item "L" in the ATIS domain. Seen in a number of different database fields, it can variously denote limousine availablity, lunch service, or other classes of service available on a flight. Yet in the following example it is clear that its usage is relevant to just the first of these:

What is transport code L?

Our claim is that it is not just the referent of "L"—limousine service for ground transportation—that plays a role here, but also the means by which it gets to that referent: namely, by being an abbreviation or code rather than a name. That is, "transport code L" is a meaningful compound, while "transport code limousine" would not be. The lexical entry for abbreviation terms like "L" reflects this by taking the form of an inverse function application: the referent of the lexical item "L" is the ground transportation type that has the string "L" as its abbreviation:

```
(INVERSE-VAL* (ABBREV-OF) "L" (GROUND-TRANSPORTATION))
```

While this has the same referent as the lexical entry for "limousine" it has a different form, one which the rule analyzing the construction above makes use of.

Nominal compounds in the DELPHI system are generated by the following rule:

```
(N-BAR :NOM3)
→
(N :NOM2)
(N-BAR :NOM1)
{NOM-COMP-READING :NOM1 :NOM2 :NOM3}
```

in which the constraint relation NOM-COMP-READING computes the semantics of the whole construction from the semantics of the head noun (passed up to N-BAR via the variable `:NOM1`) and of the nominal modifier (`:NOM2`). NOM-COMP-READING has different solutions for different semantic types of noun translation. The relevant one here is:

```
(NOM-COMP-READING
     (NOM :PARS1 (INVERSE-VAL* :FUNCTION :TERM1 :SET) :ARG-SORT)
     (NOM (CONS-P (REL1) (:ARG type :ARG-SORT) :PARS2)
          (REL-APPLY* :FUNCTION :NP)
          :VAL-SORT)
     (NOM :PARS1 (INVERSE-VAL* :FUNCTION :TERM1 :SET) :ARG-SORT))
→ ∅
```

The first slot of the NOM terms above encodes the argument-taking property of relational nouns such as "code", "salary" or "speed", and has been described above. The rule states that an inverted attribute reference (here, "L") preceded by a relational noun (here, "code") for that same attribute (here, "ABBREV-OF") simply refers to that inverted attribute reference.

Our view is that the preceding nominal modifier essentially performs a function of disambiguation: it serves to distinguish the desired sense of the head from any other possible one. This is reinforced when the whole compound—"transport code L"—is considered. Another NOM-COMP-READING rule is responsible for combining "code" with "transport":

```
(NOM-COMP-READING
     (NOM :PARS1 :ARG-SET :ARG-SORT)
     (NOM (CONS-P (REL1) (:ARG type :ARG-SORT) :PARS2)
          (REL-APPLY* :FUNCTION :NP)
          :VAL-SORT)
     (NOM (CONS-P (REL1) :ARG :PARS2)
          (REL-APPLY* (F-RESTRICT :FUNCTION :ARG-SET) :NP)
          :VAL-SORT))
→ ∅
```

This constrains the domain of the relational noun it modifies to be just the set that is the translation of the modifying noun. The semantic type of the modifying noun must be unifiable with the argument type of the head relational noun. After this unification forms the translation of the compound "transport code", the resulting type constraints serve to distinguish the correct sense of "L"—that of ground transportation—from the meal service and other senses.

Our technique of allowing rules to impose restrictions on the forms of semantic translations themselves, rather than merely on the semantic types of this translations, bears some discussion because it differs from proposals made by others, such as [66]. In that work, the position is taken that inspecting or restricting the structure of a logical form is inadmissible on theoretical grounds, in that it violates or makes unenforceable the principle of compositionality. As far as theoretical matters go, we believe that the principle of compositionality is open to many interpretations (for example, see [72]) and that its most general interpretation does not exclude techniques such as ours.

A more practical concern, and one which may well underly or justify the theoretical qualm, is that rules based on the structure of logical form may not succeed if that structure happens to be transformed (say through wrapping with another structure) into some syntactically different form which cannot be recognized as one which the rule should allow.

This is not a problem for the rules presented in this section, since the operation of nominal compounding is so tightly localized, operating in a function-argument fashion that gets to the noun meaning "first", before other modifications such as postnominal adjuncts. Ultimately, though, we feel that the real solution must lie in a different approach to meaning representation, one in which non-denotational properties of the utterance meaning are encoded or highlighted in a way that stands above the variances of syntactic logical form.

3.6.3 Impact of Our First Implementation of Constraint Relations on Parsing

Constraint relations are generally useful in that they allow one to give a name to a particular condition and use it in multiple places throughout the grammar. Consider verbs which take PPs and ADJPs as complements. In "John became happy", it is intended that that the adjective "happy" apply to the subject "John". It would not make sense to say "The table became happy". Similarly, in "I put the book on the floor", the PP "on the floor" is intended to apply to the object NP "the book" and it would not make sense to say "I put the idea on the floor" Semantic type constraints in such cases clearly hold not just between the verb and its various arguments, but *between the arguments themselves*. A constraint relation like PREDICATIVE-PP can be used to express this relationship between arguments where it is needed.

The core of the DELPHI parser is a bottom-up left-to-right algorithm, discussed in more detail in Chapter 4. Formally speaking, this algorithm can parse the kind of grammar we have been discussing without any modification, since the constraint relations and their solutions can simply be incorporated into the algorithm's empty symbols table.

For a non-toy domain, however, this increases the size of the parse tables intolerably. For our first implementation of constraint relations for semantic interpretation, therefore, we modified the algorithm so that it treated empty constraint relation symbols specially, not expanding them when the parse tables were built but instead waiting until parse time where it solved them top-down through a process that might be thought of as a kind of all-paths non-backtracking Prolog.

A problem still arose when constraint relations received traces as arguments. Until a trace is bound, it of course contains very little information, and hence unifies with almost any constraint relation solution. Since bottom-up parsing often hypothesizes traces, there is a consequent combinatorial explosion which can lead to slow parsing.

The obvious solution to this problem was simply to defer the attempt to solve constraint relations until the point in the parse where they have received adequate instantiation. The definition of "adequate" clearly differs from constraint relation to constraint relation: in the case of PREDICATIVE-PP it might be that the preposition and class of NP object be known. Until constraint relations are sufficiently instantiated to bother solving, they can simply be carried as extra riders on chart edges, being passed up as new edges are built.

This solution, too, was inadequate, since it required either manually marking the constraint relations whose solutions needed to be deferred or delaying the solution of all constraint relations. Our ultimate solution to the problem of using constraint relations was two-fold:

- Restructuring the grammar where possible so that PPs and similar modifiers and their associated constraint relations were introduced at a point in which all the information available for a solution was available. Our use of recursive structures in NP (Section 2.3) is an example of one such restructuring.
- Utilizing general interpretation principles that would only attempt to construct a semantic formula for a given constituent at a point in the parse when all necessary information was available. The mapping units and general labelled argument mechanisms discussed in Sections 3.4 and 3.5 are examples of this.

```
  TRANSLATE-TAXONOMY(top) ::=
[

  CONJUNCTION-CLASSES := REGULARIZE(top)

  TRANSLATIONS := TRANSLATE(top)

  (for pairing in CONJUNCTION-CLASSES
    do  tmp := :ANY
       (for class in pairing[2]
            do tmp := UNIFY(TRANSLATIONS(class),tmp))
       TRANSLATIONS(pairing[1]) := tmp)

 TRANSLATIONS
]

TRANSLATE(concept) ::=
 [
   DISJOINTNESS-CLASSES :=
       (PICK-ARBITRARY-ORDER
           (SET s (POWER (CHILDREN concept))
               (AND (NON-EMPTY s)
                    (FORALL x s (FORALL y s (-> (NOT (= x y))
                                                 (DISJOINT x y)))))))

  (for class in DISJOINTNESS-CLASSES
    do (for sub-concept in class
          do (for trans in (TRANSLATE sub-concept)
                do (TRANSLATIONS trans[1]) :=
                       (UNIFY (CONS concept
                                     (for class' in DISJOINTNESS-CLASSES
                                        collect (if (= class class')
                                                    trans[2]
                                                    :ANY)))
                              (TRANSLATIONS trans[1])))))

   TRANSLATIONS(concept) :=
        (CONS concept (for class in DISJOINTNESS-CLASSES collect :ANY))

   TRANSLATIONS
  ]
```

Figure 3.8: Translation Algorithm

Chapter 4

Parsing

In this chapter we examine the changes in the processing strategies for using grammatical knowledge in DELPHI. We first begin by examining the introduction of procedural elements into our grammatical formalism, to increase efficiency. Next, we describe the use of statistical training techniques to transform our parser from all-paths to best-first. Finally, we detail the use of fragment-processing and frame-based fall-back techniques to handle ill-formed and extra-grammatical utterances. We can look at these techniques as maximizing the performance of the system along the following line:

- procedural elements increase the efficiency of the use of the grammar as a whole
- statistical techniques optimize the use of the most common rules of the grammar
- fallback strategies allow the system to handle extra-grammatical constructions, whether they are due to limited coverage, ill-formedness, or recognition errors

4.1 Introducing Procedural Elements into Unification Parsing

Unification grammars based on complex feature structures are theoretically well-founded, and their declarative nature facilitates exploration of various parsing strategies. However, a straightforward implementation of such parsers can be painfully inefficient, exploding lists of possibilities, and failing to take advantage of search control methods long utilized in more procedurally-oriented parsers. In the context of BBN's DELPHI system, we have explored modifications that gain procedural efficiency without sacrificing the theoretical advantages of complex feature-based grammars.

One class of changes was to introduce varieties of structure sharing or "folding" to control combinatorics. One kind of sharing was achieved automatically be partially combining similar grammar rules in the tables used by the parser. Another resulted from introducing a strictly limited form of disjunction that grammar writers could use to reduce the number of separate rules in the grammar.

The other class of changes introduced procedural elements into the parsing algorithm to increase execution speed. The major change here was adding partial prediction based on a procedurally-tractable and linguistically-motivated subset of grammar features. Appropriate choice of the features on which to base the prediction allowed it to cut down substantially the space that needed to be searched at runtime. During this first phase of our work, we also explored the use of non-unification computation techniques for certain subtasks, where the nature of the computation is such that approaches other than unification are significantly faster but can still be integrated effectively into an overall unification framework.

Together, the classes of changes discussed here resulted in up to a 40-fold reduction in the amount of structure created by the parser for certain sentences and an average 5-fold parse time speedup in the BBN DELPHI system.

4.1.1 Folding of Similar Structures

Major improvements were obtained by partially combining similar structures, either automatically, by combining common elements of rules, or manually, through the use of a limited form of disjunction.

Automatic Rule Folding

The goal here was to provide an automatic process that would take advantage of the large degree of similarity frequently found between different rules in a unification grammar by overlapping their storage or execution paths. While the modularity and declarative nature of such a grammar are well-served by representing each of a family of related possibilities with its own rule, storing and testing and instantiating each rule separately can be quite expensive. If common rule segments could be automatically identified, they could be partially merged, reducing both the storage space for them and the computational cost of matching aginst them.

We implemented a scheme that combines rules with equivalent first elements on their right hand sides into rule-groups. This kind of equivalence between two rules is tested by taking the first element of the right hand side of each rule and seeing if they mutually subsume each other, meaning that they have the same information content. If so, the variables in the two rules are renamed so that the variables in the first elements on their right hand sides are named the same, meaning that those elements in the two rules become identical, although the rest of the right hand sides and the left hand sides may still be different. This equivalence relation is used to divide the rules into equivalence classes with identical first right hand side elements.

Before this scheme was adopted, the parser, working bottom up and having discovered a constituent of a particular type covering a particular substring, would individually test each rule whose right hand side began with a constituent of that category to see if this element could be the first one in a larger constituent using that rule. For example, each of the dozen VP rules the grammar contains that begin with a V would have to be separately matched. (In the earlier version of the grammar, before the mapping unit approach to subcategorization was adopted, this would have been on the order of 80 VP rules.) After adding rule-groups, the parser only needs to match against the common first right side element for each group. If that unification fails, none of the rules in the group are applicable; if it succeeds, the resulting substitution list can be used to set up continuing configurations for each of the rules in the group. Use of the rule-groups scheme collapsed over 270 rules out of the 453 of the full grammar into approximately 70 rule-groups, meaning that each single test of a rule-group on the average did the work of testing between three and four individual rules. Some rule groups were much larger, however, such as those whose left most constituent was V (12), DETERMINER (12), or NP (19).

Note that the kind of efficiency added by this use of rule-groups closely resembles that found in ATN's, which can represent the common prefixes of many parse paths in a single state, with the paths only diverging when different tests or actions are required. But because this process here occurs as a compilation step, it can be added without losing any of the original grammar's modularity and clarity. While similar goals could also be achieved by rewriting the grammar, for example, by creating a new constituent type to represent the shared element, such changes would make the grammar less perspicuous and less easy to extend. Under our regime, the grammar writer is left free to follow linguistic motivations in determining constituent structure and the like, while the system take care of restructuring the grammar to allow it to be executed more efficiently.

Limited Disjunction

While in some circumstances it seems best to write independent rules and allow the system to discover the common elements, there are other cases where it is better for the grammar writer to be able to collapse similar rules using disjunction. That explicit kind of disjunction, of course, has the same obvious advantage of allowing a single rule to express what otherwise would take *n* separate rules, which would have to be matched against separately and which could add an additional factor *n* ambiguity to all the structures build by the parser. For example, the agreement features for a verb like "list" used to be expressed in BBN's DELPHI system using five separate rules:

```
(V (AGR (1ST) (SNG)) ...) → (list)
(V (AGR (2ND) (SNG)) ...) → (list)
(V (AGR (1ST) (PL)) ...) → (list)
(V (AGR (2ND) (PL)) ...) → (list)
(V (AGR (3RD) (PL)) ...) → (list)
```

That information can be expressed in a single disjunctive rule:

```
(V (:OR (AGR (1ST) (SNG))
        (AGR (2ND) (SNG))
        (AGR (1ST) (PL))
        (AGR (2ND) (PL))
        (AGR (3RD) (PL))) ...) → (list)
```

Many researchers have explored adding disjunction to unification, either for grammar compactness or for the sake of increased efficiency at parse time. The former goal can be met as in Definite Clause Grammars [74] by allowing disjunction in the grammar formalism but multiplying such rules out into disjunctive normal form at parse time. However, making use of disjunction at parse time can make the unification algorithm significantly more complex. In Karttunen's scheme [52] for PATR-II, the result of a unification involving a disjunction includes "constraints" that must be carried along and tested after each further unification, to be sure that at least one of the disjuncts is still possible, and to remove any others that have become impossible. Kasper [54], while showing that the consistency problem for disjunctive descriptions in NP-complete, proposed an approach whose average complexity is controlled by separating out the disjunctive elements and postponing their expansion or unification as long as possible.

Rather than pursuing efficient techniques for handling full disjunction within unification, we have taken quite a different tack, defining a very limited form of disjunction that can be implemented without substantially complicating the normal unification algorithm. The advantage of this approach is that we already know that it can be implemented without significant loss of efficiency, but the question is whether such a limited version of disjunction will turn out to be sufficiently powerful to encode the phenomena that seem to call for it. Our experience seems to suggest that it is.

Much of the complexity in unifying a structure against a disjunction arises only when more than one variable ends up being bound, so that dependencies between possible instantiations of the variables need to be remembered. For example, the result of the following unification

```
?AGR ⊓ (:OR (AGR (2ND) (SNG))
       (AGR (2ND) (PL))
       (AGR (3RD) (PL)))
```

(where "?" marks variables and "⊓" means unification) can easily be represented by a substitution list that binds **?AGR** to the disjunction itself, but the following case

```
(AGR ?P ?N) ⊓ (:OR (AGR (2ND) (SNG))
                   (AGR (2ND) (PL))
                   (AGR (3RD) (PL)))
```

requires that the values given to **?P** and **?N** in the substitution list be linked in some way to record their interdependence. In particular, it seemed that if we never allowed variables to occur inside a disjunction nor any structure containing more than one variable to be matched against one, then the result of a unification would always be expressible by a single substitution list and that any disjunctions in that substitution list would also be only disjunctions of constants. Thus we required that disjuncts contain no variables, and that the value matched against a disjunction either be itself a variable or else contain no variables.

However, enforcing the restriction against unifying disjunctions with multi-variable terms turns out to be more complex than first appears. It is not sufficient to ensure, while writing the grammar, that any element in a rule that will directly unify with a disjunctive term be either a constant term or a single variable, since a single variable in the rule that directly matches the disjunctive element might have already, by the operation of other rules, become partially instantiated as a structure containing variables, and thus one that our limited disjunction facility would not be able to handle.

For example, if the disjunctive agreement structure for "list" cited above occurs in a clause with a subject NP whose agreement is **(AGR (3RD) (PL))**, and in a containing VP rule that merely identifies the two values by binding them both to a single variable, the conditions for our constrained disjunction are met. However, if the subject NP turns out to be pronoun with its agreement represented as **(AGR ?P ?N)**, the constraint is no longer met.

This problem with our fast but limited disjunction turned up, for example, when a change in a clause rule caused agreement features to be unbound at the point where disjunctive matching was encountered. The change was introduced to allow for queries such as "Do United or American have flights ... ", where the agreement between the conjoined subject and the verb does not follow the normal rules. The solution to that problem, as discussed above in Section 2.2 was to introduce a constraint node [20] pseudo-constituent, placed by convention at the end of the rule, to compute the permissible combinations of agreement values, with the values chained from the subject through this constraint node to the VP. Unfortunately, because of the placement of that constraint node in the rule, this meant that the agreement features were still unbound when the VP was reached, which caused our disjunctive unification to fail.

In our current implementation, the grammar writer who makes use of disjunction in a rule must also ensure that the combination of that rule with the existing grammar and the known parsing strategy will still maintain the property that any element to be unified with a disjunctive element will be either a constant or a single variable. Mistakes result in errors flagged during parsing when such a unification is attempted. We are not happy with this limitation. Nevertheless, the result of our work so far has been a many-fold reduction in the amount of structure generated by the parser without any significant increase in the complexity of the unification itself.

4.1.2 Search Space Reduction

A second class of changes in addition to those that fold together similar structures are changes in the parsing algorithm that reduce the size of the search space that must be explored. The major type of such control was a form of prediction from left context in the same sentence.

Prediction

A form of prediction for context free grammars was described by Graham, Harrison, and Ruzzo [39] that was complete in the sense that, during its bottom-up, left-to-right parsing, their algorithm never tries to build derivations at word w using rule R unless there is a partial derivation of the left context from the beginning of the sentence up to word $w - 1$ that contains a partially matched rule whose next element can be the root of a tree with R on its left frontier. This is done by computing at each position in the sentence the set of non-terminals that can follow partial derivations of the sentence up to that point.

While this style of prediction works well for context free grammars, simple extension of that method to complex feature-based grammars founders due to the size of the prediction tables required, since a separate entry in the prediction tables needs to be made not just for each major category, but also for every distinct set of feature values that occurs in the grammar. One alternative is to do only partial prediction, ignoring some or all of the feature values. (The equivalent for context free grammars would be predicting on the basis of sets of non-terminals.) This reduces the size of the prediction tables at the cost of reducing the selectivity of the predictions and thus relatively increasing the space that will have to be searched while parsing.

The amount of selectivity available from prediction based on particular sets of feature values depends, or course, on the structure of the particular grammar. In the BBN DELPHI system, we found that prediction based on major category only, ignoring all feature values, was only very weakly selective, since each major category predicted almost the full set of possible categories. Thus, it was important to make the prediction sensitive to at least some feature values. We achieved the same effect by splitting certain categories based on the values of key features and on context of applicability. Prediction by categories in this adjusted grammar did significantly reduce the search space.

For example, our original grammar used the single category symbol S to represent matrix (or root) clauses, subordinate clauses, and relative clauses. This had the effect on prediction that any context which could predict a sub-clause ended up predicting both S itself and everything that could begin an S, which meant in practice almost all the categories in the grammar, since virtually any category can appear as the initial constituent of a matrix clause. By dividing the S category into three—ROOT-S for matrix clauses, REL-S for relative clauses, and S for all other subordinate clauses—we were able to limit such promiscuous prediction. Furthermore, the distinction between root and non-root clauses is well established in the linguistic literature (see Edmonds [36]), and relative clauses are known to behave differently than other subordinate clauses with respect to various extraction phenomena [79, 80]. Having this distinction encoded via separate category symbols, rather than through subfeatures, allows us to more easily separate out the phenomena that distinguish the three types of clauses. For example, root clauses express direct questions, which are signalled by subject-aux inversion ("Is that a non-stop flight?") while subordinate clauses express indirect questions, which are signalled by "if" or "whether" unaccompanied by subject-aux inversion ("I wonder if/whether that is a non-stop flight.") And relative clauses and indirect questions use different forms of WH-pronouns; relative clauses also allow for forms without an overt WH pronoun at all, which is impossible in any sort of question, direct or indirect. Thus it seems that the distinction needed for predictive precision at least in this case was also one of more general linguistic usefulness.

We can look at the division of the former S category into ROOT-S, REL-S, and S as a species of hand-compilation. It should be pointed out that the division of S into three subtypes has another small effect on efficiency. Before the division, both matrix, relative, and subordinate clauses had to carry a full range of features for mood, tense, trace, etc., some of which are predictable for root clauses and which need not be tested against. With the division of S into ROOT-S, REL-S, and S, many of the former features have been removed from ROOT-S and REL-S, requiring fewer unifications to be performed whenever a clausal utterance or relative clause is built, which is quite often. Here are the current specifications of the features for these three categories:

```
(ROOT-S COMPFLG WH CONJFLAG)
(REL-S SONS-P TRACE CONJFLAG)
(S COMPFLG MOOD WH TRACE TRACE CONJFLAG)
```

All three categories contain **CONJFLAG**, which indicate whether the clause is a conjunction or not. **COMPFLG** indicates what complementizer introduces the clause. For ROOT-S, this can either be null or "what if" for conditionals. For S, it can either be null or "that" for indirect statements, or some WH pronoun or "if" or "whether" for indirect questions. This feature must be passed up for ROOT-S, in order to derive the correct semantic interpretation; and for S, since different verbs selecting clausal complements put different constraints on their complements (e.g. indirect question or statement, "that" required or optional for indirect statements, etc.) REL-S does not require this feature at all, since its introductory material is limited to the relative complementizers "that" and null and the relative pronouns, and is not externally selected at all. **WH** appears on both ROOT-S and S, since they may be either statements or questions, but does not appear on REL-S, since it has no option. REL-S contains a single **TRACE** flag, to bind the relativized position within the clause to the head of the relative, while S has two, to provide the correct links between extracted WH phrases and their traces via the well-known difference list [73] or "trace-threading" [24] mechanism. ROOT-S has no need for this feature, since, by definition, there is no place external to a matrix clause to bind a trace. Finally, S contains a **MOOD** feature to allow clausal complement taking constituents to select the mood of the clause; contrast "I think that he is arriving" with "I demand that he arrive on time". ROOT-S and REL-S only take the indicative form, so there is no need for them to bear this feature.

Another category which we found useful to split was VP. As was mentioned in [24], VP in our former grammar was used to handle the following types of constituents:

- a verb and its complements: "What time does the 7:20 AM flight *arrive in Dallas?*"
- an infinitival complements: "I need *to arrive in Baltimore before 9 o'clock.*"
- an imperative clause: "*Show flights first class on American Airlines between Dallas and Philadelphia.*"

In our earlier report, we had discussed assigning different constituent labels to these three constructions, reserving VP for a verb and its complements. During this project, we carried this plan out, and now have the three categories:

VP for a verb and its complements.

COMPL-VP for infinitival complements.

TOP-VP for imperatives.

VP and COMPL-VP share the same set of features, while TOP-VP has a much smaller set, since it does not need to agree in features with any dominating constituent.

We also made possessive pronouns, such as "my" and "her", members of the category POSSESSIVE-PRONOUN; previously, they had been NPs.

A further major gain in predictive power occurred when we made linguistic trace information usable for prediction. The presence of a trace in the current left context was used to control whether or not the prediction of one category would have the effect of predicting another, with the result of avoiding the needless exploration of parse paths involving traces in contexts where they were not available.

Like the rule-groups described earlier, prediction brings to a bottom-up unification parser a kind of procedural efficiency that is common in other parsing formalisms, where information from the left context cuts down the space of possibilities to be explored. Note that this is not always an advantage; for parsing fragmentary or ill-formed input, one might like to be able to turn prediction off and revert to a full, bottom-up parse, in order to work with elements that are not consistent with their left context. However, it is easy to parameterize this kind of predictive control for a unification parser, so as to benefit from the additional speed in the typical case, but also to be able to explore the full range of possibilities when necessary. In fact, we have done just this in the implementation of the syntactic fragment combiner discussed below in Section 4.3. When prediction fails in fallback mode, we revert to a full, bottom-up parse, based on the constituents found: essentially, we scan from left to right, looking for elements such as prepositions and determiners that tend to be good indicators of the left margin of a constituent. We then proceed, returning to our use of prediction, with possible reinvocation of bottom-up information if we cannot find any constituent(s) that span the remainder of the input while remaining consistent with left-context.

Non-Unification Computations

We also integrated non-unification computations into the parser, where these can still provide the order-independence and other theoretical advantages of unification. There are some subproblems that need to be solved during parsing that do not seem well-suited to unification, but for which there are established, efficient solutions. As was noted in Section 2.2.1, as our technique for handling such problems, we have implemented a mechanism that allows us to write constraint nodes to be compiled into LISP code. This ability is crucial to the greater part of the work we have done to change our grammar from a fairly straightforward implementation of Definite Clause Grammar to a device for transducing between written or spoken input. Without the ability to produce certain solutions via computation, we would have been unable to implement the labelled argument architecture describe in Sections 2.4 and 3.4.

4.2 The Use of a Statistical Agenda in Parsing

4.2.1 The GHR Algorithm

In a Graham/Harrison/Ruzzo (GHR) parser [39], such as was used in the initial implementation of DELPHI [24], a chart is used to maintain a record of syntactic constituents that have been found (terms) and grammatical rules that have been partially matched (dotted rules). Parsing strategies such as GHR, CKY [53] and other algorithms can be viewed as methodical ways of filling the chart which are guaranteed to explore all possible extensions of dotted rules by terms.

An agenda is an alternative chart-filling algorithm with the goal of finding some term covering the entire input without necessarily filling in all of the chart. If terms can be ranked by "goodness" and the grammar can produce multiple analyses of a given string, then one goal for an agenda is to produce the "best" parse first.

In order to speed up processing time we have chosen to use the agenda mechanism to reduce the search necessary to produce *acceptable* (see below) parses. This results in sparsely populated charts, approaching the extreme (and probably unattainable) goal of deterministic parsing, in which the only terms and dotted rules entered into the chart are those which appear as parts of the final parse.

The techniques involved in statistical agenda parsing allow "low probability" rules to be added to a grammar without significant cost in terms of either erroneous parses or increased parse time. These low probability rules greatly increase the coverage and robustness of the system by accounting for unusual or marginal constructions.

4.2.2 DELPHI Agenda Parsing

Most techniques for search space reduction involve careful tuning of the grammar or the parsing mechanism. This is very labor intensive and can place limits on the grammatical coverage of the system [1]. Our approach is to use an automated statistical technique for ranking rules based on their use in parsing a training set with the same grammar (under the control of an all-paths GHR parser without human supervision).

This approach also allows us to include grammatical rules that are of use only rarely, or in specialized domains, and to learn how applicable they are to a body of sentences. To take into account general linguistic tendencies, we augment the statistical ranking by a small number of general agenda ordering strategies.

The DELPHI agenda mechanism is based on three "schedulable" action types:

1. the insertion of a term into the chart,
2. the insertion of a dotted rule into the chart, and
3. the (conditional) "pair extension" of a dotted rule by a term.

In principle one would like to order those actions in terms of the probability that they lead to a final parse. The initial implementation of the agenda mechanism uses an approximation to this ordering.

4.2.3 Use of Statistical Measures

There are two types of measures that one might estimate to help the agenda parsing mechanism. They are (1) category expansion probabilities and (2) rule success probabilities.

Category Expansion Probabilities

Category expansion probabilities are perhaps the more obvious of the two measures. The goal is to determine the probability that a given syntactic category (e.g., NP) is expanded by a given grammar rule in a valid parse.

These probabilities allow one to estimate the probability that a given tree is the expansion of a given category. Bayes' rule may be used to calculate the relative probabilities of various parse trees for a specified input string.

Rule Success Probabilities

Using rule success probabilities, the goal is to determine the probability that a term inserted into the chart by a particular rule will be part of a final parse.

Training

In order to train the agenda mechanism, a set of sentences is parsed using the all-paths GHR parser and their charts are analyzed.

For each rule (R) in the grammar we determine three numbers:

1. $NT(R)$, the number of terms in the charts based on that rule.
2. $NDR(R)$, the number of dotted rules initiated in the chart based on that rule.
3. $NGT(R)$, the number of "good terms" based on that rule, ones that are constituents of an *acceptable* parse (e.g., ones leading to executable database commands for ATIS).

   For each category C in the grammar, we calculate one number:

4. $NGT(C)$, the number of terms with that category which are constituents in an acceptable parse.

The ratio $\frac{NGT(R)}{NT(R)}$ is an estimate of the probability that a term based on R will appear in the final parse, and $\frac{NGT(R)}{NDR(R)}$ is an estimate of the probability that the initiation of a dotted rule based on R will lead to a good term. (Note that in DELPHI, each word sense is treated as if it were a separate grammar rule, and so this mechanism takes into account the relative likelihood of various word senses in the training set.)

If $C(R)$ is the category produced by the rule R, then the category expansion probability of R is $\frac{NGT(R)}{NGT(C(R))}$.

Preliminary Results for Different Measures

Using rule success probabilities leads to substantial reduction (a factor of more than 3) in chart size. In general, one might expect that better estimates of such probabilities, based on category expansion probabilities in the tree below the term, would lead to improved results, even though these estimates require somewhat more computation than rule success probabilities alone.

We have compared the use of category expansion probabilities with the use of rule success probabilities in several variations of the agenda mechanism, and have found that rule success probabilities produce superior results, although the reasons for this are not entirely clear.

An experiment using category expansion probabilities alone led to larger charts than produced by the use of rule success probabilities in isolation. Combining category expansion probabilities with rule success probabilities appeared to be no better than just using the rule success probabilities.

4.2.4 Agenda Structures

The structure of the agenda mechanism appears to be as important as the statistical measures used to order agenda items. Experience with probabilistic agendas in speech processing would suggest an approach in which all information relevant to ordering is combined into a single numeric measure and used to order a single queue. In principal, this allows different measures to interact and for strength in one measure to make up for weakness in another.

We experimented with this approach in a system which had a single agenda in which all three of the schedulable action types described above were placed. The statistical measures described above were combined in a weighted fashion with priorities based on the size of the constituents, the position of the right hand end of the constituent and the action type. A number of experiments were run, giving different weightings to the different parameters, but all of these experiments led to charts that were 20% to 40% larger than the alternative structured agenda described below.

The structured-agenda approach involves the creation of a 2-dimensional array of agendas, as illustrated in Table 4.1.

	Rightmost Endpoint				
Action Type	N	$N-1$	...	1	0
Pairs	A_1	A_4	...		
Rules	A_2	A_5			
Terms	A_3	A_6			

Table 4.1: Agenda

Each cell of the array consists of a single type of action, e.g. term insertion, and all of the actions in the list A_i in a cell have the same rightmost end. Within the cell, the actions in the list A_i are ordered by probability estimates.

For each step, the first non-empty cell (starting with A_1 and going in the order shown in Table 4.1) is chosen, and the first item on its agenda is run. This has the effect of reinforcing progress to the right through the input string, of choosing the most appropriate action for such motion at each step, and favoring close attachment of modifiers.

4.2.5 DELPHI Results

Measurements of chart-size and time reductions for BBN's DELPHI grammar running on the ATIS0 and ATIS1 training and test sets indicate the improvements possible with several variations of the basic agenda mechanism. For example, using the structured agenda on the 551 ATIS0 sentences of training data from June 1990, the chart size was reduced by a factor of 3.24, and the total processing time reduced by a factor of 1.82.

This result underestimates the improvement gained by agenda parsing, since somewhat more than 10% of the "sentences" in the training data were ill-formed according to our grammar (many were ill-formed according to any plausible grammar!). Since a properly operating agenda system will eventually produce the same chart that the GHR parser does, and since that entire chart must be searched before a string is determined to be unparseable, the performance of any agenda mechanism must reduce to that of the GHR parser for such inputs.

Another set of experiments was performed with a set of 539 "parseable" strings taken from the combination of the ATIS0 (June 1990) and ATIS1 (February 1991) training sets. For this set the speedup was a factor of 3.8 and the chart size reduction was well over 3.5. (The hedge on chart size reduction is because data for the chart size of 5 sentences in the GHR parser was not obtained, the charts overflowed available memory . At this time the ratio of that chart size to the size of the agenda parser chart was over 30.)

The introduction of probabilistic agenda parsing, combined with the application of software engineering techniques, has sped up natural language analysis considerably. The average time for parsing, semantic interpretation, and discourse processing in our original implementation was lowered to 1.43 seconds per sentence, with a median time of 0.99 seconds, on a Sun 4/280. The most recent version of our algorithm has comparable or even faster times, and includes not just the natural language processing stages, but is an actual "end to end" figure including retrieval of an answer from the application database.

4.3 The Use of Fragment Processing

In this section, we describe the fallback understanding component of DELPHI. This component is invoked when DELPHI's regular chart-based parser is unable to parse an input; it attempts to come up with a parse and semantic interpretation, or a semantic interpretation alone, based on a fragmentary analysis of the input.

The fallback understanding component consists of three separate stages, which are invoked successively. First, the Fragment Generator produces a sequence of fragmentary sub-parses from the chart state left over from the unsuccessful parse. Next, two different combination modules—the Syntactic Combiner and Frame Combiner—employ alternative and complementary strategies for combining these fragments. This is shown in the diagram in Figure 4.1.

The Syntactic Combiner uses extended grammar rules that can skip over intervening material to combine constituents in an attempt to re-construct a plausible parse of the input. This parse can be a clause or some other useful constituent such as an imperative VP. A semantic interpretation for this reconstructed parse is automatically provided through the action of the grammar rules.

The Frame Combiner is invoked when the Syntactic Combiner is unsuccessful. It utilizes a set of domain dependent pragmatic slot-filling schemata that embody the goals that users most commonly have; for example, in the ATIS domain, such tasks as finding a flight or fare that satisfies some set of constraints, or asking about ground transportation between an airport or a city. As such, it determines only a semantic interpretation and not a parse.

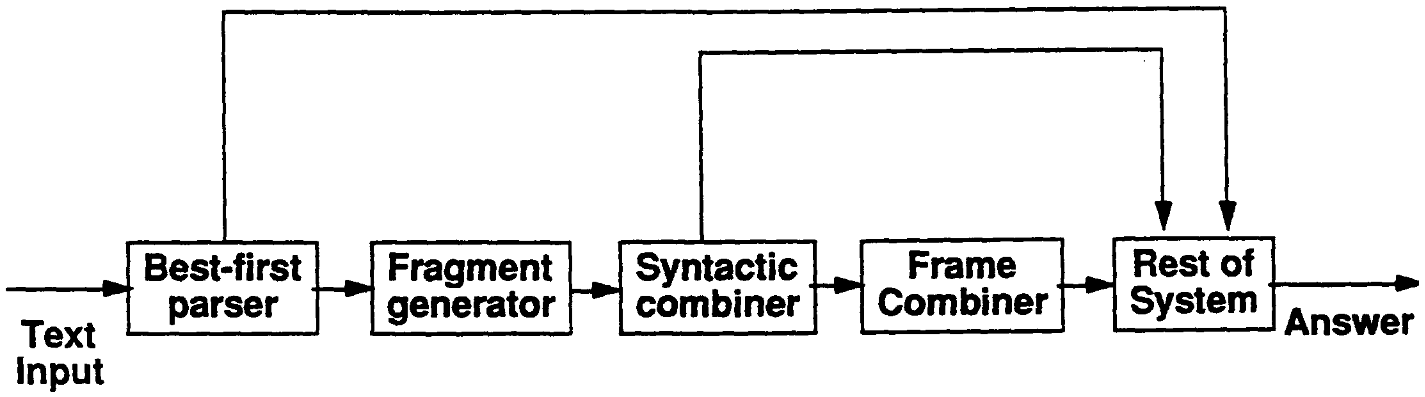


Figure 4.1: Fallback Processing Architecture

The intent of this multi-step approach to fallback processing is to provide a smoother path between the accuracy but fragility of regular parsing on the one hand, and the robustness but possible inaccuracy of schemata-based methods on the other.

The remainder of this section is taken up with detailed description of each component. We describe the Fragment Generator, Syntactic Combiner, and the Frame Combiner in detail. Finally, we present the February 1992 NL and SLS evaluation test results for these components, separate and combined.

4.3.1 The Fragment Generator

Recall that DELPHI's grammar rules incorporate semantic constraint and interpretation components by associating with each element of the right-hand side a grammatical relation label which keys into an associated system of semantic rules. This feature means that any term which is inserted into the chart is guaranteed to be semantically well-formed and to be annotated with one or more semantic interpretations.

The fragment generator generates a set of such semantically annotated fragments from the chart state left over after an unsuccessful parse. The algorithm for generating fragments from the chart extracts the most probable terms associated with longest sub-strings of the input, using probabilities associated with the producing rules in the manner describe in the preceding section.

For example, the utterance:

I want a flight uhh that arrives in Boston let's say at 3 pm

is conventionally unparseable due to the interpositions "uhh" and "let's say". The Fragment Generator produces the following set of four fragments:

```
S[I want a flight]
NO-INTERP[uhh]
REL-S[that arrives in Boston]
NO-INTERP[let's say]
PP[at 3 pm]
```

4.3.2 The Syntactic Combiner

The Syntactic Combiner uses a special set of grammar rules, called fragment rules, to combine these fragments into a single parse. These rules have the same form as rules of the regular DELPHI grammar and incorporate semantic constraints and interpretation rules in the same way. But the method for applying the fragment rules differs in that it allows them to combine constituents even when these constituents are separated by intervening portions of the input, or when they occur in a reversed order.

Each fragment rule is adjunction oriented, in the following form:

```
X → :head X, :other-relation C
```

The following is an example, from which unification features have been omitted:

```
VP → :head VP, :pp-comp PP
```

This rule says that an existing Verb Phrase fragment and an existing Prepositional Phrase fragment can be combined together to make a new Verb Phrase with the original VP as head and the original PP as pp complement, provided they satisfy the semantic constraints associated with "`:head`" and "`:pp-comp`".

The central operation of the Syntactic Combiner is adjunction. The example rule licenses the Syntactic Combiner to "adjoin" one fragment tree into another—that is to replace a node of the first tree with a new node whose head daughter is the old node and whose other daughter is second tree. An example, using the rule above, would be the combination of the two fragments:

```
PP[at 3 pm]
VP[arrives in Boston]
```

to make the new VP:

```
VP[arrives in Boston at 3 pm]
```

Note that the adjunction node does not have to be the top of the first fragment tree: it can be any non-terminal node, as in the following pair of fragments:

```
PP[at 3 pm]
S[NP[I]
  VP[want
     NP[NP[a flight]
        REL-S[that
               VP[arrives in Boston]]]]]
```

The algorithm that applies these rules first scans right to left taking each successive fragment and looking for fragments to its left to adjoin the first fragment into. The search for an attachment point within a fragment is right-to-left, bottom-up first, and deterministic.

The reason for the directional priority is to enforce the preference of fragment rules that the sub-term of the adjunction be to the right of the head. The algorithm then reverses direction, attempting to adjoin any remaining fragments into other fragments on their right. It oscillates back and forth in this fashion until no more fragments can be combined.

At the end of this process the largest fragment (possibly now containing other fragments which it has absorbed) is returned as the reconstructed parse, subject to cut-off restrictions which we discuss below. More than one fragment is returned in the case of multiple clausal fragments, and the discourse module is invoked to construct the interpretation of the whole.

As a simple example, let us return to the example of the previous section:

I want a flight uhh that arrives in Boston let's say at 3 pm

which generates the fragments:

```
S[I want a flight]
NO-INTERP[uhh]
REL-S[that arrives in Boston]
NO-INTERP[let's say]
PP[at 3 pm]
```

The rules that enable combination of these fragments are:

```
VP→ :head VP, :pp-comp PP
NP→ :head NP, :rel-clause REL-S
```

The first rule above licenses the attachment of "at 3 pm" to "arrives" inside the existing REL-S structure "that arrives" and the second the attachment of the combined REL-S structure to the NP "a flight" inside the clause "I want a flight". After this combination, we are left with two fragments: a clause and an unanalyzable portion of the string. Since all grammar rules in DELPHI include a semantic interpretation component, as discussed in Chapter 3, a semantic interpretation of the clause is also available.

The other fragment rules combine NPs and their various modifiers and VPs and their NP complements:

```
NP→ :head NP, :pp-comp PP
NP→ :head NP, :post-nom NP
NP→ :head NP, :whiz-rel VP
VP→ :head VP, :direct-object NP
```

The Syntactic Combiner uses a cut-off (currently .8) for the ratio of the number of words included in the final reconstructed parse to the number of words of the original input to determine whether or not to accept the final analysis as plausible. The computation of this ratio is adjusted to ignore a stop list of certain words that carry little meaning ("does" "me" "could" etc.) and to block interpretations which exclude other words which do tend to change the meaning ("first", "most" etc.).

4.3.3 The Frame Combiner

The Frame Combiner seeks to combine together not fragments but the semantic interpretations of fragments, and does so based not on grammar rules but on pragmatic schemata which have various "slots" to fill. It works primarily with semantic interpretations of fragments at the NP and PP level. Its approach is similar in spirit to SRI's Template Matcher [47] but it differs from that work in a number of important ways.

Most importantly, it is fully integrated with a conventional NLU system including grammar and parser. This makes it possible for it to work from recursive tree fragment structures instead of sub-strings of input. As a result, the slot-filling process is not limited to simple phrases such as "to BWI" but can also handle more syntactically and semantically complex phrases such as "to the airport closest to Washington DC". All the complex modifier structure internal to NPs which a conventional parser normally uncovers can be incorporated into slot-filling.

Moreover, while the system does not use larger constituents such as VPs and clauses to fill slots directly, it does make use of a candidate NP or PP's occurrence *inside* such a larger phrase to determine which slot the candidate should fill. This enables the Frame Combiner to cope with such cases as the PP "before 3 pm", which means entirely different things, and therefore constraints entirely different slots, depending on whether it modifies the verb "arrive" or "depart".

A final difference is that the Frame Combiner attempts to determine the actual items of information that the user wants to have presented to him—that is, what slots in the frame are being asked about, as opposed to filled or constrained. This last has practical importance within the context of the ATIS task domain because it enables only what is asked about to be displayed to the user. Formerly it was sufficient simply to provide the entire extension of a suitably frame as the answer, but given the MIN/MAX scoring procedure, such a tactic is likely to result in numerous wrong answers.

The basic operation of the Frame Combiner is to input a sequence of semantically annotated fragment trees and to output a logical form as a proposed interpretation of the utterance. As intermediate steps it generates alternative sets of attribute-value "triples" and filters these according to plausibility criteria before generating a final interpretation from the chosen set. We next describe each of these steps.

Representational Triples

As intermediate output, the frame combiner first produces a set of attribute-value triples with the following structure:

<OPERATOR, ATTRIBUTE, VALUE>

The *ATTRIBUTE* is a single or multi-valued function. The *VALUE* is an element or set of elements from this function's range. The *OPERATOR* is a binary relation over elements of the range. In the following example:

<EQUAL, ORIGIN-CITY, BOSTON>

The operator is the relation *EQUAL*. The attribute in this example is the function *ORIGIN-CITY*, whose domain is the class *FLIGHT* and whose range is the class *CITY*. The value in the example is the individual city *BOSTON*.

Other typical operators are relations like *TIME-BEFORE* and *GREATER-THAN*. There is a special operator, *HAS-PROPERTY*, which is combined with a truth-valued (i.e. one-place predicate) attribute and the value *TRUE* for adjectival meanings like "non-stop".

Currently there are three classes which can serve as the domain of an attribute—*FLIGHT*, *FARE* and *GROUND-TRANSPORTATION*. We refer to these as the "core" classes of the ATIS task. These core classes are associated, respectively, with the distinguished attributes *FLIGHT-OF*, *FARE-OF* and *TRANS-OF*, which we term the "explicit" attribute of the core class. Explicit attributes are necessary to incorporate well-formed, parsed NP fragments whose semantic type is one of the core classes, such as "the USAir flight from Boston to Denver", without having to break them up into their component modifiers. Explicit attributes are always combined with the EQUAL operator and an element of the domain, and effectively correspond to the identity function for the domain.

An attribute-value triple can be formally viewed as a specification of a subset of the domain of the attribute of the triple. While they have a clear relationship to the notion of a template or frame, they are perhaps better thought of as disembodied "slot-constraints". Note in particular that a set of such triples is a more flexible representation than a single template in that it can uniformly combine triples whose attributes have different domains. This is important when the question itself concerns more than one domain—such as both *FLIGHT*s and *FARE*s.

Generating Triples

Triples are produced from fragment trees using a recursive-descent algorithm that applies a set of pattern rules that match against fragment trees and their attached semantic interpretations. Rules can produce disjunctions of triples in case of ambiguity. The rules primarily match against NP and PP constituents, associating the semantic interpretation of the NP constituents with the value element of a triple. The algorithm mainly recurses through other types of constituents, though it does note and pass down certain items of information associated with them, such as the head-predicate of a VP.

Rules consist of a syntactic pattern component followed by optional extra constraints and an attribute assignment component. For example the rule:

```
(PP :pp FROM :object)
→
(SORT :object CITY)
(RESTRICT-SLOT EQUAL (:OR ORIGIN-CITY TRANS-TO-CITY) :object)
```

applies to PPs whose preposition is "from". It requires that the NP object of the PP be of the semantic class **CITY**. It restricts either the **ORIGIN-CITY** or **TRANS-TO-CITY** attributes to be **EQUAL** to the semantics of the NP object. When applied to the fragment:

```
[PP from
    [NP boston]]
```

it generates the following two triple alternatives:

<EQUAL ORIGIN-CITY BOSTON>
<EQUAL TRANS-TO-CITY BOSTON>

corresponding to the two alternatives possible in a situation where "from Boston" is uttered: either the user wants to fly from Boston to some different city or he wants to get from Boston to its airport.

Rules have a slightly more complicated form when they involve an important feature of the Frame Combiner's triple-generation process: its use of syntactic structure and context.

For example, in the ATIS domain the PP "at 3 pm" means something very different when attached to a verb like "arrive". This phenomenon tends to pose a problem for conventional non-integrated template matching system, as has been noted in earlier work [47].

In the Frame Combiner this is handled by passing down the predicate representing a verb's meaning as an extra argument to the recursive descent algorithm. If a constituent was attached to a VP with a particular meaning, the slot-filling process knows this when it reaches the constituent. Slot-filling rules can be written in such a way as to behave differently depending on whether the constituent under consideration is in the context of a particular verbal predicate.

For example, in order to deal with the above phenomenon, the following rule applies to PP fragments where the NP **:OBJECT** is of type **TIME-OF-DAY**, and **:PREP** is any preposition from which a temporal relation can be derived. This temporal relation restricts whatever slot is determined appropriate by the **ATTRIBUTES** component of the rule:

```
(PP :PP :PREP :OBJECT)
→
(SORT :OBJECT TIME-OF-DAY)
(TEMPORAL-RELATION :PREP :REL)
(RESTRICT-SLOT
   :REL
   (ATTRIBUTES
     (CONTEXT ARRIVE ARRIVAL-TIME)
     (DEFAULT DEPARTURE-TIME)
     (GENERAL ARRIVAL-TIME DEPARTURE-TIME))
   :OBJECT)
```

The **ATTRIBUTES** expression delivers zero or more attributes as a disjunction of the specific attributes depending upon which of its evidence clauses is satisfied. **CONTEXT** evidence is the strongest. **NON-LOCAL** evidence is next, and it includes situations where a particular verb is merely present elsewhere in the input, without dominating the constituent. **DEFAULT** evidence is the assignment preferred whereas **GENERAL** evidence is all the assignments allowed.

Filtering Sets of Triples

When all fragments have been analyzed through recursive descent, the system takes the Cartesian product of all disjunctive interpretations to obtain the set of all alternative sets of triples. These are then filtered to leave only the most plausible sets of triples.

There are several criteria for plausibility. The most obvious is that two or more triples on the same attribute not specify contradictory values for the attribute. Another is that a set not contain any two triples with clashing attribute domains. For example, in the ATIS task one never sees queries that combine flights and ground transportation (even though such are certainly expressible, e.g. "Show me USAir flights to airports that have limousine service"). Thus *FLIGHT* and *GROUND-TRANSPORTATION* are clashing domains. On the other hand, queries concerning both flights and fares do frequently occur ("Show flights to Boston and their fares") so *FLIGHT* and *FARE* are not clashing domains.

Another criterion is that the set of triples have the commonly seen linguistic form for the domain. Thus, while "the airport" and "the city" are plausible fillers for **`TRANS-TO-AIRPORT`** and **`TRANS-TO-CITY`** in the *GROUND-TRANSPORTATION* domain they are much less plausible fillers for *FLIGHT* domain attributes, simply because proper noun fillers are far more common for these.

Criteria such as non-clashing domains are hard criteria, and therefore any triple set which violates them is discarded. Other criteria, like the plausibility of linguistic domain, are softer, and the system merely prefers not to violate them.

If there is more than one plausible set of triples, the Frame Combiner will, depending on switch setting, either give up or appeal to extrasentential discourse to resolve the ambiguity (much as the core DELPHI system does).

Choosing the Information to Display

At each turn in dialogue, any system performing an information retrieval task, such as ATIS, is essentially required to display a set of objects. This holds for WH questions ("which flights..."), imperatives ("show me"), and existential yes-no questions ("are there any flights..."). On this perspective, the different sets of objects and relationships between them are one part of the meaning of the query, and are represented by the sets of triples. The other part of the meaning is the question of which of these sets to display. We refer to this as the "topic" of the query.

To choose one (or more) of the triples as the topic means to display its value set, as it relates to all other value sets of the other triples. Several different heuristics are used, and are ranked in priority. Each is tried in succession until a topic is chosen.

Most obvious is whether the filler of the triple is a WH noun phrase. If it is, it definitely must be the topic.

Next are any "priority" domains that are not normally used merely to constrain other sets. An example is *GROUND-TRANSPORTATION*—the typical ATIS user does not ask to see cities that have a particular type of ground transportation—the user wants to see the ground transportation itself.

"Unconstrained" triples are another likely topic. A triple is "unconstrained" if its filler is a bare common nominal, such as "airline", and its attribute is a total function. Since every *FLIGHT* has an **AIRLINE**, the user is most unlikely to be imposing the vacuous constraint that the flight is on some airline (even though this is again expressible). Rather, the user is much more likely to be interested in seeing the airline of the flight.

Generating the Final Interpretation

The Frame Combiner generates a final logical form from a chosen set of triples by first associating a variable with each triple filler ("value" slot) and a variable with each of the core classes present, in the set, whether through explicit attributes on the class or implicitly as the domain of another attributes. It generates a matrix formula in which all the attributes present are binary relations and the generated variables are the arguments to these binary relations. Quantificational structure, corresponding to the fillers of triples, is then generated. The quantifiers for topic triples are treated as though they were WH quantifiers, and appropriate display commands generated.

4.3.4 Results and Discussion

As an attempt to measure the effects of these different fall back strategies, we ran a number of tests using the February 1991 cross-site evaluation test data. Using the same constant executable Lisp image ("disksave") run for the official results, the test was run using a number of different switch settings, and scored with the version of the NIST comparator used for the official results. The switch conditions were: no fallback processing at all (which is simply the core DELPHI system), Syntactic Combiner only, Frame Combiner only, and both Syntactic Combiner and Frame Combiner working together (which was the condition used in the official results). The figures for NL only are reported in Table 4.2.

Note the frame-only condition is actually better than result officially reported, in which both fallback sub-components were used.

For the SLS test, the output of BBN's BYBLOS N-best recognizer was used, with N = 5. The core DELPHI system (without fragments) was first tested against the five theories.

	%T	%F	%NA	%WE
no fallback	69.3	7.4	23.3	38.1
syn only	73.1	8.9	18.0	35.8
frame only	78.3	9.6	12.1	31.3
both(official)	76.7	10.6	12.7	33.9

Table 4.2: NL Results, February, 1992

If an interpretation was found for one of them, it was returned. Otherwise, the fallback methods were applied.

Results for three of the four conditions are seen in Table 4.3 (results for the no-fallback = core DELPHI condition were unavailable as of this writing). The figure for the combination of both fragment modules (the configuration used in the official test) reflects an slight downward adjustment from the original value of 43.7 that corrects a purely procedural error committed during our running the test (the file that specifies "today's date" for each query was not loaded, leading to a small increase in the number of wrong answers). This problem was fixed in obtaining the results in Table 4.3.

	%T	%F	%NA	%WE
syn only	68.9	11.5	19.7	42.6
frame only	73.8	13.0	13.2	39.2
both(official)	71.8	15.4	12.8	43.7
both(adjusted)	71.9	15.1	13.0	43.2

Table 4.3: SLS Results, February, 1992

As in the regular NL test, the SLS results show an noticeable improvement over the official results when the Frame Combiner is used alone.

These results tend to undercut a central premise of our original strategy: namely that using both fragment combination methods together would improve the result over the use of either alone. Our tentative hypothesis is that the Syntactic Combiner, when failing and passing to the Frame Combiner the best results of its combination attempt, is passing wrongly combined fragments which mislead the Frame Combiner.

On the other hand, these results do show the utility of the Frame Combiner when used alone. For NL only, it reduced the No Answer rate by 11.2 percentage points, and Weighted Error by 4.2 percentage points. For SLS, it reduced the Weighted Error from the adjusted official value of 43.2% to a new low of 39.2%.

Chapter 5

Discourse

During the course of this project, we implemented the first discourse component for DELPHI. In this chapter, we discuss the current discourse module of DELPHI (Section 5.1), which is designed for the ATIS domain, whose special discourse requirements are detailed. We also discuss the initial implementation in DELPHI of domain independent discourse techniques, based on earlier work done at BBN for other DARPA contracts [8] (Section 5.2).

5.1 Current Discourse Module

5.1.1 Discourse Phenomena in the ATIS Domain

We begin our discussion of DELPHI's current discourse module with an overview of the type of phenomena found in the ATIS common task domain.

While the first tests of natural language capability in the DARPA SLS program stressed the ability to understand single utterances, later tests have taken into account the requirement that a natural language system used for the ATIS task is intendeded to facilitate conversations involving more than a single utterance. The speaker typically builds up a travel plan over the course of several interactions with the system, and the information already given for the plan is assumed and is usually not repeated nor explicitly referred to by the user in later utterances. This implies that natural language systems for problem solving tasks such as ATIS must be sensitive to the context of the task and the conversation, in order to interpret sentences as the user meant them.

The following sequences of queries from actual problem solving sessions from the ATIS2 corpus indicate some of the phenomena that must be covered. DELPHI currently handles nearly all of these examples correctly. Numbers in parentheses are the official utterance IDs assigned by NIST.

The first session shows a type of anaphoric reference that is quite standard in the linguistic literature—"that flight" (in e70073sx) refers back to the flight mentioned in the preceding sentence (e70073sx).

(e70073sx) What's the latest flight out of Denver that arrives in Pittsburgh next Monday?
(e70083sx) What's the economy class fare for that flight?

The second session shows a number of phenomena. The first sentence (0h0016vx) is literally a description of the user's goals, but the system must interpret it as a command to display the acceptable flights. The second sentence (0h0026vx) is clearly an elliptical utterance, intended to add a constraint to the previous sentence. The third sentence (0h0036vx) shows an interesting and frequent characteristic of the ATIS domain. What looks like a clear definite reference "flight number seven thirty one" must be treated in the context of the dialogue, since there are often several flights with the same number. In fact, this must be taken in the context of the answer to the previous query, and not the text of the previous queries alone. This is particularly true since connecting flights are represented with strings like "US258/US424" and users refer to such flights with varied expressions. Among those we have seen are:

flight two fifty eight
US flight two five eight
US Air two five eight slash four two four
US two five eight four two four

Note that "US flight two five eight" may refer to the connection US258/US424, or to the first flight in the connection, and that a single flight such as US258 can go from one city to a second and on to a third, and thus the reference can be intended to any of the legs or the entire flight with that identifier.

(0h0016vx) I would like to fly from Boston to Philadelphia next Thursday.
(0h0026vx) Between three and four PM.
(0h0036vx) What type of aircraft is flight number seven thirty one?

The third session shows a deictic reference to "today" (e70011sx), which must be resolved by the system based on the reported date of the interaction. The second sentence (e70021sx) shows "one" used anaphorically, and the third (e70031sx) shows a deictic anaphor which can be interpreted in terms of either the previous query or its answer.

(e70011sx) Give me the flights from Boston to San Francisco leaving early today.
(e70021sx) Show me the one that arrives the earliest in San Francisco.
(e70031sx) What kind of plane is that?

The fourth session shows the type of "constraint extension" dialogue that is common in this domain. The second utterance (1s0023vx) is certainly a sentential (clausal) utterance (perhaps missing an origin or destination phrase), not a Noun Phrase or a Prepositional Phrase. It must be taken as a further constraint on the trip plan that the user has indirectly mentioned in the first query. The seemingly "obvious" pronominal reference "it" in the third sentence (1s0033vx) does not refer to any Noun Phrase in the preceding sentences, but rather to the flight implicitly specified by the second sentence. A clearer case of elliptical specification of further constraints is given in the fifth utterance (1s0053vx). More interestingly, the fourth utterance (1s0043vx) *resets* the assumed constraints given previously, otherwise the fifth utterance could not be interpreted correctly as specifying a flight from Boston to Atlanta (not the reverse) leaving (this is the default assumption for times in these tasks, judging by user's actions) at 8:24 PM.

(1s0013vx) Do you have a flight from Atlanta to Boston?
(1s0023vx) I would like to go at six thirty six AM.
(1s0033vx) Where does it stop?
(1s0043vx) I would like to fly from Boston to Atlanta.
(1s0053vx) Eight twenty four PM.

The fifth session shows more constraint extension. It raises the question as to when a constraint context is reset. In most cases when the user explicitly mentions an origin and destination, all previous constraints are likely to be dropped. In the last utterance (1r0043vx), it is unclear whether the date constraint "(On) august thirty first" (from 1r0023vx) should still be in force.

(1r0013vx) Okay, I'd like to fly from Denver to Pittsburgh.
(1r0023vx) The date will be August thirty first.
(1r0033vx) I'd like to go in the morning.
(1r0043vx) Is there a flight from Denver to Pittsburgh around eight AM?

The last session illustrates that a discourse in the ATIS domain can switch between topics: here, from flight planning to ground transportation planning and back again.

(ew0013ss) What flights leave from Boston to Pittsburgh in the morning?
(ew0023ss) Is there anything that arrives before eight fifty seven?
(ew0033ss) How about on Sunday night?
(ew0043ss) How do I get from Pittsburgh airport to downtown Pittsburgh?
(ew0053ss) What are the rates for the ground transportation?
(ew0063ss) Are there any flights that arrive before nine PM on Sunday?

5.1.2 A Discourse Module for ATIS

Most previous work on discourse context for natural language interfaces has focused on recovering meaning from context in the face of explicit linguistic cues in an utterance. Such cues include pronouns and definite references (e.g. "Which of *them* stop in Dallas", "Do any of *those flights* serve breakfast"), as well as clearly elliptical utterances (e.g. "*on Sunday*", "*morning flights*", "*round trip*"). The DELPHI system includes a general domain-independent mechanism for dealing with such phenomena.

The initial version of the Discourse Module utilized the notion of "discourse entities", discussed in more detail in Section 5.2 below, which represent those entities which are introduced either explicitly or implicitly in previous utterances, and which can be referred to by pronouns and definite noun phrases. It included mechanisms for resolving references to those entities, including use of syntactic constraints to limit possible intra-sentential referents, and a focusing model for dialogue state to limit and direct references to entities in past utterances. This work is based on earlier work done at BBN in other DARPA contracts [8].

Later work focused on handling the implicit discourse phenomena such as constraint extension that seems to be critical in the ATIS domain. This was done in a separate component of the system we refer to as the Task Tracker.

In the last year of the contract, it became clear that linguistic issues were only part of the problem in end-to-end evaluations, and that a noticeable loss of performance occurred in mapping from the semantic representation produced by the semantic interpreter and that used by the database retrieval component. Thus, as part of our move to the use of generalized grammatical relations (Section 3.5), which involved a major change to the semantic interpreter, we decided to implement a more robust quantification module that produced semantic representations of the same type as used by the back-end. Unfortunately this meant that the implementation of the Discourse Module (described in the next section), which utilized our previous form of semantic representations, could no longer be used, since the discourse entity code was no longer functional. We believe, however, that the theory and algorithms still are the appropriate basis for a more general discourse component.

An analysis of the training data convinced us that it would be more effective and efficient to implement a much simpler interim pronominal mechanism, and focus on the semantic interpreter and the task tracker, with the intent of reimplementing the full discourse entity and focusing model once the remainder of the system had stabilized. The remainder of this section describes the stages of implementation of the Discourse Module, including the discourse entity/focusing/communicative act Discourse Module, which has been temporarily replaced in the final system produced under this contract (but which will be reintroduced in later systems).

5.2 Initial Implementation

In order to treat multi-sentence discourse phenomena, we ported the discourse module used in the Janus multi-modal interface system to DELPHI[8]. This component maintained a representation of discourse state composed of "communicative act" structures representing each user input, machine action (response), or other communicative occurrence. Things that could be referred to are encoded as discourse entities, elements created at the level of discourse representation which are hypothesized on the basis of syntactic information without being limited to syntactic constituents [96][97]. For example, consider the fragment of a dialogue:

Q1: What flights depart from Boston for San Francisco after 7 PM?
A1: ...
Q2: Show me their fares.

Note that "their" in Q2 refers to "flights from Boston to San Francisco after 7 PM" even though there is no single syntactic constituent of this form in Q1. However, a corresponding discourse entity containing just this information is produced on the basis of the information in Q1.

In order to eliminate spurious candidates as the antecedents of pronouns, we use syntactic constraints on intra-sentential anaphora, such as those described in [46] and [24], as part of our anaphora resolution mechanism. Centering algorithms [27][41][89][90] are used to track focus for extra-sentential anaphora. Porting to DELPHI involved interfacing the discourse component to DELPHI's meaning representation and parse structures. A new component for handling referent ambiguity was added which presented the options to the user.

After the initial port, we added several new capabilities to our discourse module. Among them was a facility for head-noun and noun-phrase ellipses. An example of the latter is the following dialogue:

Q1: What airlines fly to Washington?
A1: ...
Q2: Dallas?
A2: ...

Here the second question, though not a complete sentence, is understood to be a shorthand for "What airlines fly to Dallas?" Our discourse module was enhanced to handle such ellipses.

We also added a capability for handling definite references. Definite references are Noun Phrases such as "this flight" and "the fares" that are intended to refer to a specific entity or a group of entities. Our system uses the semantic class information present in the Noun Phrase to search for an entity in the preceding discourse that the definite reference refers to. In the case of a definite reference that contains an open slot to be filled—i.e. the head noun is a relational noun (see Section 3.6.1)—such as "the cost", the system looks for an entity in the preceding discourse that can fill the slot.

5.2.1 Modifications for ATIS

When the ATIS domain was chosen for the purposes of cross-site evaluation, we began analyzing the discourse phenomena that occur in that domain. Although the general-purpose discourse module of DELPHI already covered most instances of explicit reference, it needed to be extended to obtain good coverage in ATIS.

Previous work on discourse context has focused on recovering meaning from context in the face of explicit linguistic cues in an utterance. Such cues include pronouns and definite references (e.g. "Which of *them* stop in Dallas", "Do any of *those flights* serve breakfast"), as well as clearly elliptical utterances (e.g. "*on Sunday*", "*morning flights*", "*round trip*"). The DELPHI system includes a general domain-independent mechanism for dealing with such phenomena.

As we saw in Section 5.1.1, an investigation of the transcripts for ATIS showed that the context dependence in this task was not always indicated by such explicit linguistic cues. Rather, a travel plan is gradually built up by the speaker, and the information already given for the plan is assumed and not repeated nor referred to explicitly by the user.

Since the user of an ATIS system is typically planning a flight, which is a "constraint satisfaction" type of task, a standard strategy is for the user to set up a discourse context that specifies some constraints, such as "I need to fly from Boston to Dallas on Monday." Later interactions assume that all such constraints are maintained unless explicitly or implicitly modified or removed. Thus, even though a sentence like "Are there any nonstop flights?" looks on the surface like it is interpretable independent of the discourse context (cf. a sentence from the personnel database domain "Are there any Cambridge residents?"), the "flights" it refers to must satisfy the earlier constraints. In general, any reference to "flight" or "fare" in ATIS, whether a *definite* reference ("the flights"), a deictic reference ("that flight", "those flights") or an indefinite reference ("any nonstop flights") is to be assumed to meet the constraints established earlier in the discourse.

For example, in the following sequence:

List the flights from San Francisco to Atlanta, please.
Show me flights that arrive after noon.

the second sentence assumes that "flights" is already constrained to be those from San Francisco to Atlanta.

We found that of the 98 discourse pairs supplied by TI for training on the ATIS1 corpus, only 14% contained explicit reference. To increase our coverage, we began implementing an explicit representation of the "flight scenario", where sentences having to do with a flight result in the addition of constraints to the scenario. Subsequent sentences inherit those constraints. The simplest form of this approach addressed almost half of the discourse pairs.

To do this, we implemented a new module of DELPHI, the Task Tracker, which maintained a model of the user's task context as it evolves in the discourse of a dialogue. It takes as input the semantic interpretation of the query, and delivers as output a version of that semantic interpretation making explicit information that can be inferred from the Tracker's evolving knowledge of the task the user wishes to accomplish. This task tracker is intended to solve a problem similar to the one solved by the "templates" of other systems, but is more general in a number of ways.

Again, consider the following dialogue:

Q1: Show me flights from Dallas to Denver please.
Q2: Show me just the flights that arrive after noon.

the user's real intended meaning for Q2 is:

Q2': Show me flights *from Dallas to Denver* that arrive after noon.

and not simply any flight that happens to arrive after noon, no matter what its origin and destination. The Tracker infers this crucial information and adds it to the interpretation of the second query so that the correct retrieval can be made from the database. It does so by maintaining a record (the "task context") derived from the user's previous queries, and merging the stored information with the new information in the present query to find the query's intended meaning. Task contexts are frame-like, having slots for the various pieces of component information. For example, the task context for taking a flight has slots for "origin", "destination", "airline" and so forth.

To fill the slots of the flight scenario, the "task tracker" extracts constraints from the same general semantic representation that we use to represent the meaning of each utterance. The constraints are themselves represented in the same semantic representation. In the example above, we need to extract the representation for "San Francisco" and "Atlanta" to fill the "origin" and "destination" slots respectively. However, in general the filler of a slot may be a complicated expression, including other quantifiers, for example, as in "Show me flights from the airport in Boston". Thus, there is no restriction that the constraints be of a particular simple form, such as specific constant "fillers" for "slots" in a template (e.g. "Dallas" is the filler of the DESTINATION slot). We can represent constraints involving negation ("flights that do not stop in Dallas"), disjunction ("flights that stop in Dallas or Denver"), as well as more complex descriptions such as "the airport in Boston".

The original version of the Task Tracker made use of the work performed by the Discourse Module in generating discourse entities. The Task Tracker obtained the information it needed from the entities, thus minimizing the need to do manipulation of semantic representations. When the semantic representation changed, manipulation of semantic representations became much easier, but, at the same time, the discourse entity generator became unavailable, so the Task Tracker took over the job of finding the slot fillers for the scenarios.

A crucial advantage of our approach to task tracking is its modularity. The flight booking task of ATIS is in some ways a tightly constrained one, involving only a few operations such as taking a flight from one city to another, and finding a fare on a flight. By making the encoding of these constraints the responsibility of a separate module—the

Task Tracker—we can take advantage of the constraints without being bound by them in the design of the other linguistic components of our system. In addition, we built the tracker in a modular fashion, so that we could readily add new types of constraints to the tracker, and represent how the specification of a constraint could affect preexisting constraints.

The task tracker is also used in the process of interpreting single "utterances" that appear to be composed of multiple sentences, such as

(4i00b1sx) I'm sorry, I wanted to fly TWA; is there a flight between Oakland and
Boston with a stopover in Dallas/Fort Worth on TWA?

or

(4b0061sx) Find a flight between Denver and Oakland;
the flight should leave in the afternoon and arrive near five PM;
the flight should also be nonstop.

The parser and semantic interpreter break these utterances into multiple sentences, and the system treats them as a "mini-discourse", without generating a visible answer to the intermediate sentences.

Another module added to the system is a form of reference resolution that allows the user to refer to entities that appear in the answers given by the system. These are sometimes called "exaphoric references", to distinguish them from references to discourse entities that are introduced by linguistic structures in previous utterances. This is a very general phenomenon, and should in principle be taken to include such things as deictic references (e.g. "<those, these> flights", "<this, that> fare", "this" and "that"). In the current DELPHI system, however, we treat such deictic references as variants on definite references (e.g. "the flight") and rely on the task tracker to carry over the constraints necessary to specify the intended referent. There is, nevertheless, a type of exaphoric reference that is specific to the ATIS domain and which we treat separately—flight id resolution.

Flight id resolution is the process of determining the intended reference of expressions like "flight one three two" or "American one oh one". While these expressions specify properties of flights, they are generally intended to refer to specific flights, even though more than one flight may be have the given flight number and airline (the legs of a direct flight with a stop, for example). Worse, users often refer to a connecting flight by giving the airline and flight number of its first segment.

This problem is resolved when you realize that in almost all cases, the users have no idea of flight numbers until they see them on the screen as answers to previous questions.

Thus an expression involving a flight number almost always is intended as a reference to a specific flight that the system has shown to the user as part of the answer to a previous query. A general solution would involve the system keeping track of the answers it has presented (the displays it has made in the case of a graphic system), and having a mechanism to find those elements of the display which the user might have perceived as satisfying a given description. This is in general a very difficult problem, since it in principle requires having a model of how the user perceives displays and describes portions of those displays. In the ATIS domain there is a simple, and quite effective variation on this technique. DELPHI keeps track of the flights appearing in answers to previous queries, and has a mechanism to find the most recently displayed flight that satisfies the given flight id description. Every displayed flight has either one or two airline id/flight number pairs (direct flights have one pair, connections have two pairs separated by a slash). It is simple to map typical linguistic descriptions of these pairs to the pairs themselves. This mechanism as such is quite specific to the ATIS domain, but the notion of exaphoric reference resolution is quite general.

Chapter 6

Integration of Speech Recognition with NL

Many methods have been proposed for the integration of speech recognition and natural language understanding. Ideally, we would like the natural language understanding component of a spoken language understanding system to aid in the speech recognition process by providing powerful constraints on the allowable sentences. However, since the language models used for natural language understanding are usually quite complex, the tightly coupled integration of these two components usually results in very large computation.

For example, a first-order statistical language model can reduce perplexity by at least a factor of 10 with little extra computation over using no grammar, while applying complete natural language (NL) models of syntax and semantics to all partial hypotheses typically requires much more computation for less perplexity reduction. (Murveit [69] has shown that the use of an efficiently implemented syntax component within a recognition search actually slowed down the search unless it was used very sparingly.) In addition to reducing total computation, the resulting systems are more modular when we separate radically different knowledge sources (KSs).

Instead of trying to use natural language as a tightly coupled constraint on the speech recognition, we have developed the N-Best Paradigm for the integration of multiple knowledge sources. The basic idea is to use a subset of the most efficient knowledge sources to find not one, but a list of all of the likely sentence hypotheses. These hypotheses are then filtered and rescored using the remaining knowledge sources. For the problem of natural language understanding, this means that the NL component need only process the text of the most likely word sequences. Thus, the time required is reduced by several orders of magnitude. We first introduced the basic idea for this paradigm and an exact algorithm for finding the N-Best sentence hypotheses at the Cape Code DARPA Spoken Language

Workshop in October of 1989.

In the sections that follow we describe the issues and the basic algorithm in more detail below. The first section discusses the general issues of integration strategies, and introduces the N-Best Paradigm. It also includes the first exact algorithm for finding the N-Best sentence hypotheses. The second section presents two additional algorithms for finding the N-Best hypotheses. These two algorithms are approximations that require much less computation than the exact algorithm. The accuracy and speed of the three algorithms are compared. The third section presents several new uses that we have found for the N-Best hypotheses. In the fourth section, we describe how we have used the N-Best algorithm as part of our integration strategy of speech with natural language processing.

6.1 The N-Best Paradigm and an Exact N-Best Algorithm

The N-Best algorithm is a time-synchronous Viterbi-style beam search procedure that is *guaranteed* to find the N most likely whole sentence alternatives that are within a given a "beam" of the most likely sentence. The computation is linear with the length of the utterance, and also linear in N. When used together with a first-order statistical grammar, the correct sentence is usually within the first few sentence choices. The output of the algorithm, which is an ordered set of sentence hypotheses with acoustic and language model scores can easily be processed by natural language knowledge sources without the huge expansion of the search space that would be needed to include all possible knowledge sources in a top-down search.

6.1.1 Introduction

In a spoken language system (SLS) we have a large search problem. We must find the most likely word sequence consistent with all knowledge sources (speech, statistical N-gram, natural language). The natural language (NL) knowledge sources are many and varied, and might include syntax, semantics, discourse, pragmatics, and prosodics. One way to use all of these constraints is to perform a top-down tightly-coupled search that, at each point, uses all of the knowledge sources (KSs) to determine which words can come next, and with what probabilities. Assuming an exhaustive search in this space, we can find the most likely sentence. However, since many of these KSs contain "long-distance" effects (for example, agreement between words that are far apart in the input), the search space can be quite large, even when pruned using various beam-search or best-first search techniques. Furthermore, a top-down search strategy requires that all of the KSs be formulated in a predictive, left-to-right manner. This may place an unnecessary restriction on the type of knowledge that can be used.

The general solution that we have adopted is to apply the KSs in the proper order to constrain the search progressively. Thus, we trade off the entropy reduction that a KS provides against the cost of applying that KS. Naturally, we can also use a pruning strategy to reduce the search space further. By ordering the various KSs, we attempt to minimize the computational costs and complexity for a given level of search error rate. To do this we apply the most powerful and cheapest KSs first to generate the top N hypotheses. Then, these hypotheses are evaluated using the remaining KSs. In the remainder of this paper we present the N-best search paradigm, followed by the N-best decoding algorithm. We give an outline of the proof that the algorithm does, in fact, result in the correct list of sentence hypotheses. Finally, we present statistics of the rank of the correct sentence in a list of the top N sentences using acoustic-phonetic models and a statistical language model.

6.1.2 The N-best Search Paradigm

Figure 6.1 illustrates the general N-best search paradigm. We order the various KSs in terms of their relative power and cost. Those that provide more constraint, at a lesser cost, are used first in the N-best search. The output of this search is a list of the most likely whole sentence hypotheses, along with their scores. These hypotheses are then rescored (or filtered) by the remaining KSs.

Depending on the amount of computation required, we might include more or fewer KSs in the initial N-best search. For example, it is quite inexpensive to search using a first-order statistical language model, since the number of acoustic and language states is small. Frequently, a syntactic model of NL will be quite large, so it might be reserved until after the list generation. Given a list of hypothesized sentences, each alternative can usually be parsed in turn in a fraction of a second. If the syntax is small enough, it can be included in the initial N-best search, to further reduce the list that would be presented to the remainder of the KSs. We can also use this paradigm in conjunction with high-order statistical language models. While a high-order model frequently provides added power (over a first-order model), the added power may not be commensurate with the large amount of extra computation and storage needed for the search. In this case, a first-order language model can be used to reduce the choice to a small number of alternatives which can then be reordered using the higher-order model.

Besides the obvious computational and storage advantages, there are several other practical advantages of this paradigm. Since the output of the first stage is a small amount of text, and there is no further processing required from the acoustic recognition component, the interface between the speech recognition and the other KSs is trivially simple,

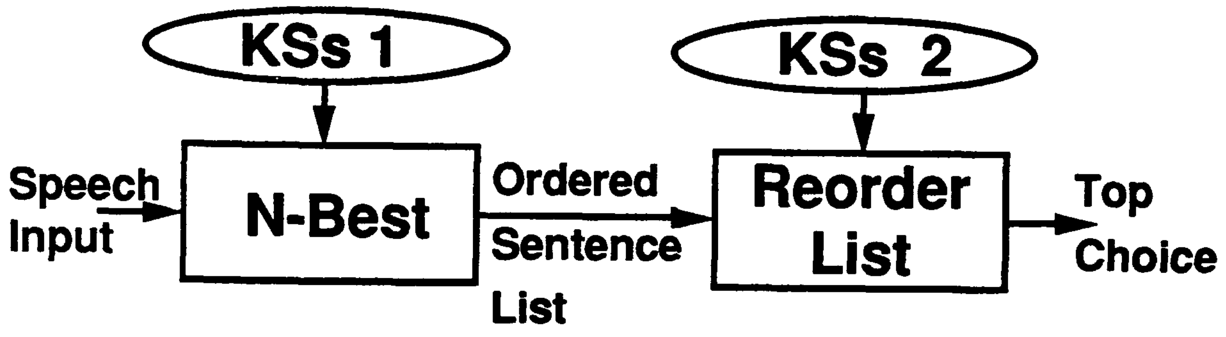


Figure 6.1: The N-best Search Paradigm. The most efficient knowledge sources, KS1, are used to find the N Best sentences. Then the remaining knowledge sources, KS2 are used to reorder the sentences and pick the most likely one.

while still optimal. As such this paradigm provides a most convenient mechanism for integrating work in a modular way. This high degree of modularity means that the different component subsystems can be optimized and even implemented separately (both hardware and software). For example, the speech recognition might run on a special-purpose array processor-like machine, while the NL might run on a general purpose host.

6.1.3 The N-Best Decoding Algorithm

The optimal N-Best decoding algorithm is, in spirit, quite similar to the time-synchronous Viterbi decoder that is used quite commonly. However, it differs in what it must compute and in its implementation. It must compute probabilities of word-sequences rather than state-sequences, and it must find *all* such sequences within the specified beam. The basic idea is to keep separate records for theories with different word sequence histories. Each path is marked with an identifier that represents the complete sequence of words up to this point (the history). When two or more paths come to the same state at the same time, we check whether there is already an existing path at that state with the same history. If there is, we add the probability for the two paths. Otherwise, we create a new path. When all paths for a state have been created, we reduce the number of paths by keeping up to a specified maximum number N of theories whose probabilities are within a threshold of the probability of most likely word sequence at that state. Note that this state-dependent threshold is distinct from and smaller than the global beam search threshold.

Since probabilities for different word sequences are kept distinct, it is easy to see that any word sequence hypothesis that reaches the end of the sentence has an accurate score. This score is the conditional probability of the observed acoustic sequence given this word sequence. Of course, since the number of possible word sequences grows exponentially, we must use a pruning algorithm to reduce it to the desired number. The interesting question is whether one can prove that all of the word sequences with probabilities greater than the threshold will end up in the list with the correct scores.

Algorithm Optimality

There have been two recent papers that deal with the topic of finding more than one answer for the whole sentence [58, 95]. However, both of these papers are based on the Viterbi algorithm. That is, when two paths for the same word sequence come to the same state, the probability is computed as the maximum of the two paths rather than the sum. Thus these algorithms find the most likely sequence of states rather than the most likely sequence of words. More importantly though, the alternative answers are constrained by the segmentation and traceback of the most likely answer. Since the segmentation of the sentence into words often depends on the words chosen, the answers found in this way are not, in fact, the best N answers. In fact, we have found in the past that this approximation is quite severe. In [95], the exact algorithm for the word sequences corresponding to the best state sequences is mentioned, but is not used, due to the computational requirements. The results given in [95] using a statistical bigram grammar of perplexity 124 show that approximately one third of the sentences that are not recognized correctly on the first choice have the correct answer within the top 10 choices found by the approximate algorithm. As will be seen in the next section, with the exact algorithm used here, for a similar statistical grammar, about 90% of the sentences that are not recognized correctly on the first choice have the correct answer within the top 10 choices found by the approximate algorithm, and about 97% are within the top 24 choices. It should be mentioned that these tests have been performed on different speech corpora, with different acoustic and language models, making direct comparisons difficult.

It should be clear that the algorithm used here would result in the exact solution for all of the possible answers for a given utterance. It is harder to see that the algorithm that finds the N-Best answers within a threshold of the best answer, in fact does so. The proof (which is not included here in its entirety) relies on the fact that the beamwidth at each state is very large—typically on the order of 10^{15}. Possible errors could occur when we should be adding two paths for the same word sequence together, but one or both of them is ignored because its score is more than 10^{15} below the best score at the state. However, if the larger of the two path probabilities was much above the threshold—say 10 times the threshold (still 10^{14} below the best score)—then the error due to ignoring the lower score is insignificant. If both are below the threshold, then when added, they can at most be twice the threshold—still quite low. Even if this happened in every frame of an utterance—an extremely unlikely event—the effect on the score would be small compared to the state beamwidth.

The result is that the algorithm will correctly detect and score all theories that are above the threshold by one order of magnitude. However, the score of theories that are within the last order of magnitude of the final beam may be slightly underestimated. This means that the state beamwidth should be one order of magnitude larger than needed, and the theories within the last order of magnitude can be ignored. When a hard limit of N is placed on the theories at each state, the effective beamwidth at that state could decrease. In this case, we must again include any theories that are within one order of magnitude below the Nth theory at the state to ensure that the final result is correct.

Implementation

This algorithm requires (at least) N times the memory for each state of the hidden Markov model. However, this memory is typically much smaller than the amount of memory needed to represent all the different acoustic models. We assume here, that the overall "beam" of the search is much larger than the "beam at each state" to avoid pruning errors. In fact, for the first-order grammar, it is even reasonable to have an infinite beam, since the number of states is determined only by the vocabulary size.

At first glance, one might expect that the cost of combining several sets of N theories (from preceding states) into one set of N theories at a state might require computation on the order of N^2. However, we have devised a "grow and prune" strategy that avoids this problem. At each state, we simply gather all of the incoming theories. At any instant, we know the best scoring theory coming to this state at this time. From this, we compute a pruning threshold for the state. This is used to discard any theories that are below the threshold. At the end of the frame (or if the number of theories gets much too large), we reduce the number of theories using a *prune and count* strategy that requires no sorting. While this would theoretically still require computation on the order of N, it only accounts for a part of the total computation. We find, empirically, that the overall computation increases with $\sqrt{N}$, or slower than linear. This makes it practical to use somewhat high values of N in the search.

6.1.4 Rank of the Correct Answer

Whether the N-best search is practical depends directly on whether we can assure that the correct answer is found reliably within the list that is created by the first stage. (Actually, if all the remaining KSs have binary scores, that is they either accept or reject a sentence, then the search is sufficient as long as there is one answer that is acceptable, since the system could never choose the lower scoring correct answer in this case.) It is possible that when the correct answer is not the top choice, it might be quite far down the list, since there could be exponentially many other alternatives that score between the highest scoring answer and the correct answer. Whether this is true depends on the power of the acoustic-phonetic models and the statistical language model used in the N-best search. Therefore we have accumulated statistics of the rank of the correct sentence in the list of N answers for two different language models: a first-order statistical class grammar (perplexity 100) [35], and no grammar (perplexity 1000). The first-order class grammar constrains the probabilities of all words in the same class to be the same, and therefore can be estimated from a small amount of training data. The experiment was performed on the speaker-dependent portion of the DARPA 1000-Word Resource Management speech corpus [76], using the BBN BYBLOS Continuous Speech Recognition System [31]. The test includes a total of 215 sentences from 12 speakers.

Figure 6.2 plots the cumulative distribution of the rank for the two different language models. The distribution is plotted for sentence N up to 100. We have also marked the average rank on the distribution. The average rank of the correct answer was 9.3 for no grammar, and the correct answer is not on the list at all about 20% of the time. However, when we use the statistical class grammar, which is a fairly weak grammar for this domain, we find that the average rank is 1.8, since most of the time, the correct answer is within the first few choices. In fact, for this test of 215 sentences, 70% of the sentences were correct on the first choice, while 99% of the sentences were found within the 24 top choices. It is also noteworthy that the acoustic model used in this experiment is an earlier version (that does not model coarticulation between words or use smoothing of poorly trained models) that results in twice the word error rate of the most recent models. This means that the likelihood that the correct answer will be found within a short list of sentences could be even higher than shown here when the better acoustic models are used.

To illustrate the types of lists that are generated we show below a sample N-best output. In this example, the correct answer is the fifth one on the list.

Example of N-best Output

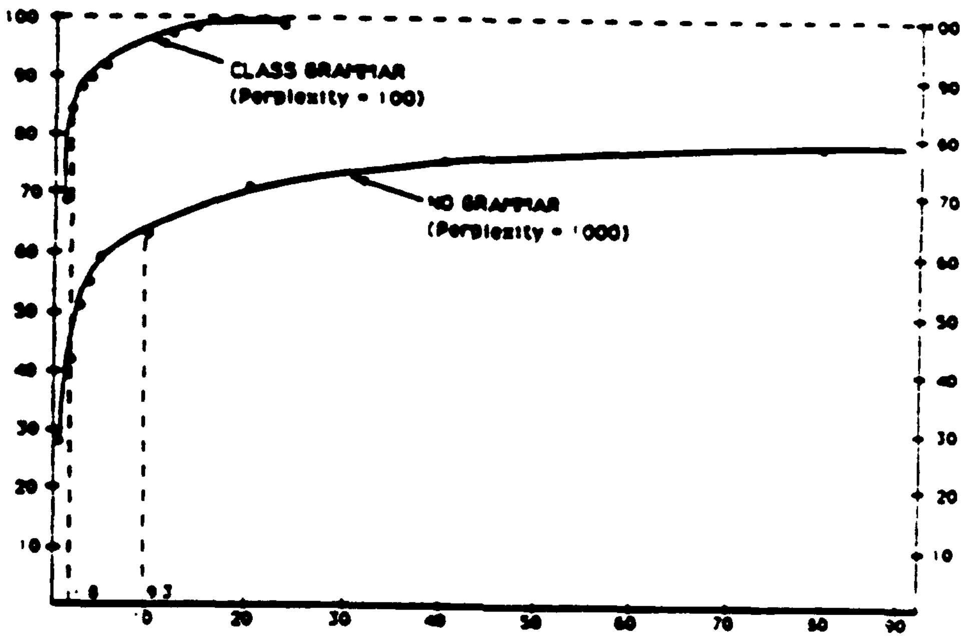


Figure 6.2: Cumulative Distribution of Rank of Correct Sentence. For the statistical class grammar, 99% of the sentences were recognized exactly within the top 24 choices.

Answer:

Set chart switch resolution to high.

Top *N* Choices:

Set charts which resolution to five.
Set charts which resolution to high.
Set charts which resolution to on.
Set chart switch resolution to five.
Set chart switch resolution to high. (***)
Set chart switch resolution to on.
Set charts which resolution to the high.
Set the charts which resolution to five.

6.2 More Efficient N-Best Algorithms

Since we introduced the N-Best Paradigm, we have invented several different algorithms for finding the N-Best sentence hypotheses. The Sentence-Dependent N-Best algorithm is an exact procedure for finding all of the sentence hypotheses whose total score is within a threshold of the most likely sentence. The computation required is linear in the number of hypotheses found. We have also developed two much faster approximate algorithms: the Word-Dependent N-Best Algorithm, and the Lattice N-Best Algorithm. The Lattice algorithm requires no more time than the usual 1-Best search, but is not as accurate as the Sentence-Dependent Algorithm. However, the Word-Dependent algorithm requires computation that is about 10 times that of the 1-Best algorithm, but is empirically as accurate as the exact method. A detailed comparison of the three algorithms and the corresponding speed and accuracy are described in more detail below.

The Word-Dependent algorithm is based on the assumption that the beginning time of a word depends only on the preceding word. We compare this algorithm with two other algorithms for finding the N-Best hypotheses: The exact Sentence-Dependent method reported in [32] and a computationally efficient Lattice N-Best method. We show that, while the Word-Dependent algorithm is computationally much less expensive than the exact algorithm, it appears to result in the same accuracy. However, the Lattice method, which is still more efficient, has a significantly higher error rate.

We also have demonstrated that algorithms that use Viterbi scoring (i.e. they find the word sequences with the most likely single state sequence) have significantly higher error rates than those that use total likelihood scoring (summed over all state sequences). (e.g. [95, 91])

The Exact Sentence-Dependent algorithm presented in Figure 6.1 is unique in that it provided the correct forward probability score for each hypothesis found. The basic idea of the algorithm is that, if two or more theories at a state involve identical sequences of words, we add their scores, since we want the likelihood of the sequence of words summed over all state sequences. Otherwise we keep an independent score for each different preceding sequence of words. We preserve all different theories at each state, as long as they are above the global pruning threshold and within the *state beamwidth* of the best score at the same state. This algorithm *guarantees* finding all hypotheses within a threshold of the best hypothesis. While the proof is not given here, it is easy to show that the inaccuracy in the scores computed is bounded by the product of the sentence length and the pruning beamwidth, which is typically a very small fraction of the score. While the number of theories that are within a threshold of the best theory at a state could theoretically grow exponentially with time (if all theories had about the same score), we find empirically that the number of theories within a threshold remains fairly constant and small. The algorithm was optimized to avoid expensive sorting operations so that it required computation that was less than linear with the number of sentence hypotheses found. In the remainder of the section, we will refer to this particular algorithm as the Exact or the Sentence-Dependent algorithm.

There is a practical problem associated with the use of this exact algorithm. In cases where we need a large number of hypotheses the computation, which is almost linear in N, becomes excessive. Frequently we compute up to 100 hypotheses for a sentence. In addition, when we examine the different answers found, we notice that many of the different answers are simple one-word variations of each other. Clearly, longer sentences would be expected to have more variations, and thus require a large value of N. Thus, much of the computation is spent finding all combinations of independent variations. In the next section, we present two algorithms that attempt to avoid these problems.

6.2.1 Two Approximate N-Best Algorithms

While the exact N-Best algorithm is theoretically interesting, we can generate lists of sentences with much less computation if we are willing to allow for some approximations. (It is important to note that, as long as the correct sentence can be guaranteed to be within the list, the list can always be reordered by rescoring each hypothesis individually at the end.) We present two such approximate algorithms with reduced computation.

Lattice N-Best

The first algorithm will derive an approximate list of the N Best sentences with little more computation than the usual 1-Best search. Within words we use the time-synchronous forward-pass search algorithm [86], with only one theory at each state. We add the probabilities of all paths that come to each state. At each grammar node (for each frame), instead of remembering only the best scoring word, we store all of the different words that arrive at that node along with their respective scores in a traceback list. This requires no extra computation above the 1-Best algorithm. The score for the best hypothesis at the grammar node is passed forward as the basis for future scoring as in the normal time-synchronous forward-pass search. A pointer to the stored list of words and scores is also sent on. At the end of the sentence, we simply search (recursively) through the saved traceback lists for all of the complete sentence hypotheses that are above some threshold below the best theory.

Figure 6.3 illustrates several alternate sentence hypotheses stored in the traceback. The locations of alternate word ends are indicated by filled circles. The resulting alternate

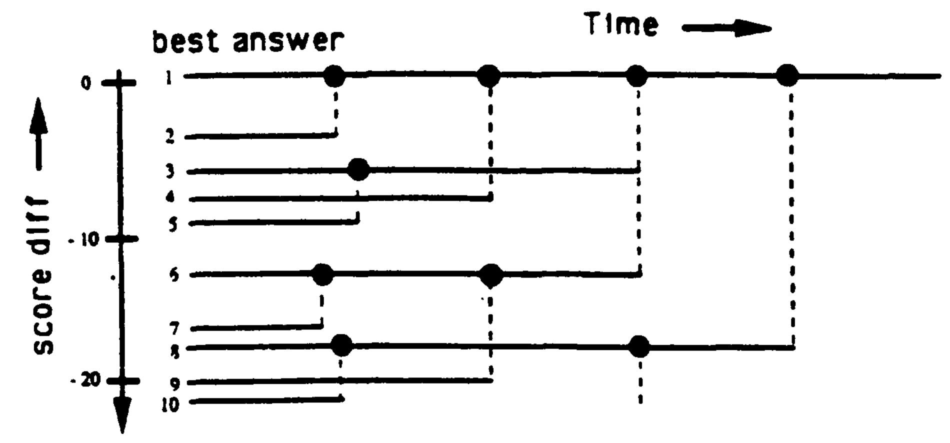


Figure 6.3: N-Best Theory Traceback. The log-score differences from the best sentence are added to compute the total sentence scores for each alternative.

sentences are shown on the left, ordered by score. Of course the number of sentences represented in this traceback lattice is huge. One algorithm for finding the most likely sentences in the traceback is presented below:

1. Initialize (clear) stack of alternate choices.
2. Initialize accumulated score decrement, s, to 0. Initialize word position, t, to the end of the sentence.
3. Perform traceback computation from t with score s, chaining back through word-ends to produce the next highest scoring sentence. Add this sentence to the list of N-Best sentence hypotheses.
4. At each word boundary along the traceback, add the accumulated score decrement, s, to each of the (negative log) differences between each alternate word score and the current sentence hypothesis. Add these differences to the stack of alternate choices.
5. Pick the alternate with the smallest difference from the best sentence.
6. Perform steps 3 and 4 (recursively) from this point in the sentence until the desired number of hypotheses has been found, or any remaining hypotheses would score too far below the best hypothesis to be of interest.

The alternate sentence hypotheses are produced in order of decreasing total score. This recursive traceback can be performed very quickly. (We typically extract the 100 best answers in a small fraction of a second.) We call this algorithm the Lattice N-Best algorithm since we essentially have a dense word lattice represented by the traceback information. An important advantage of this algorithm is that it naturally produces more answers for longer sentences, since the number of permutations of answers grows exponentially with the length of the utterance.

There is, however, a serious problem with the Lattice N-Best algorithm. It systematically underestimates or completely misses high scoring hypotheses. Figure 6.4 shows an example in which two different words (words 1 and 2) can each be followed by the same word (word 3). We assume here that the word sequence 2-3 scores better than 1-3. The dark lines for words 1 and 2 show the optimal path for each word when followed by word 3. The gray lines show two suboptimal paths for word 1. Since we allow only one theory at each state within word 3, there is only one best beginning time for word 3, determined by the best boundary between the best previous word (word 2 in the example) and the current word. But, as shown in Figure 6.4, the second-best theory involving a different previous word (word 1 in the example), would naturally end at a slightly different time. Thus, the best score for the second-best theory would be severely underestimated or lost altogether. Thus, we cannot correctly compute even the second best hypothesis without recomputing the likelihood of the different word sequence.

Word-Dependent N-Best

As a compromise between the exact sentence-dependent algorithm and the lattice algorithm, we devised the Word-Dependent N-Best algorithm. The algorithm is illustrated in Figure 6.5. We reason that, while the best starting time for a word *does* depend on the preceding word, it probably does *not* depend on any word before that. Therefore, we distinguish theories based on only the previous word rather than on the whole preceding sequence. At each state within the word, we preserve the total probability for each of $n(<< N)$ different preceding words. At the end of each word, we record the score for each previous word hypothesis along with the name of the previous word. Then we proceed on with a single theory with the name of the word that just ended. At the end of the sentence, we perform the recursive traceback (similar to that described above) to derive the list of the most likely sentences. As shown in Figure 6.5, both sequences are available at the end of word 3.

Like the lattice algorithm, the word-dependent algorithm naturally produces more answers for longer sentences. However, since we keep multiple theories within the word, we correctly identify the second best path (and all others as well).

The computation is proportional to n, the number of theories kept locally, which is typically 3 to 6. The number of local theories only needs to account for the number of pos-

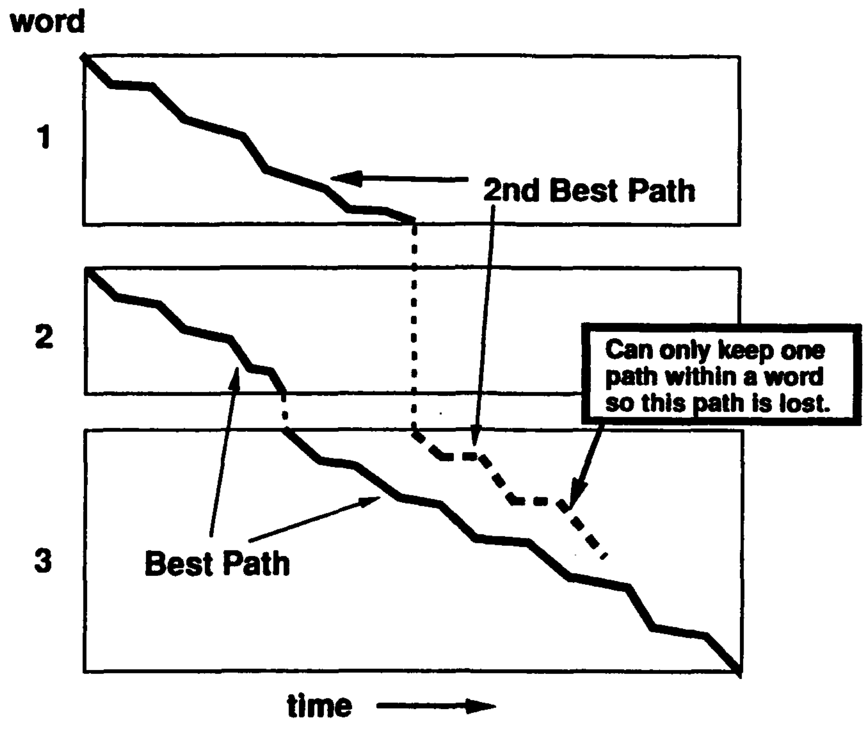


Figure 6.4: Alternate paths in the Lattice algorithm. The best path for words 2-3 overrides the best path for words 1-3.

sible previous words—not all possible preceding sequences. Thus, while the computation needed is greater than for the lattice algorithm, it is far less than for the sentence-dependent algorithm.

6.2.2 Comparison of N-Best Algorithms

We performed experiments to compare the accuracy of the three N-Best algorithms. In all cases, we used a weak first-order statistical grammar based on 100 word classes. The experiments were performed on the speaker-dependent subset of the DARPA 1000-Word Resource Management Corpus [76]. The test set used was the June '88 speaker-dependent test set of 300 sentences. The test set perplexity is approximately 100. To enable direct comparison with previous results, we did not use models of triphones across word boundaries, and the models were not smoothed. We expect all three algorithms to improve significantly when the latest acoustic modeling methods are used.

Figure 6.6 shows the cumulative distribution of the rank of the correct answer for the

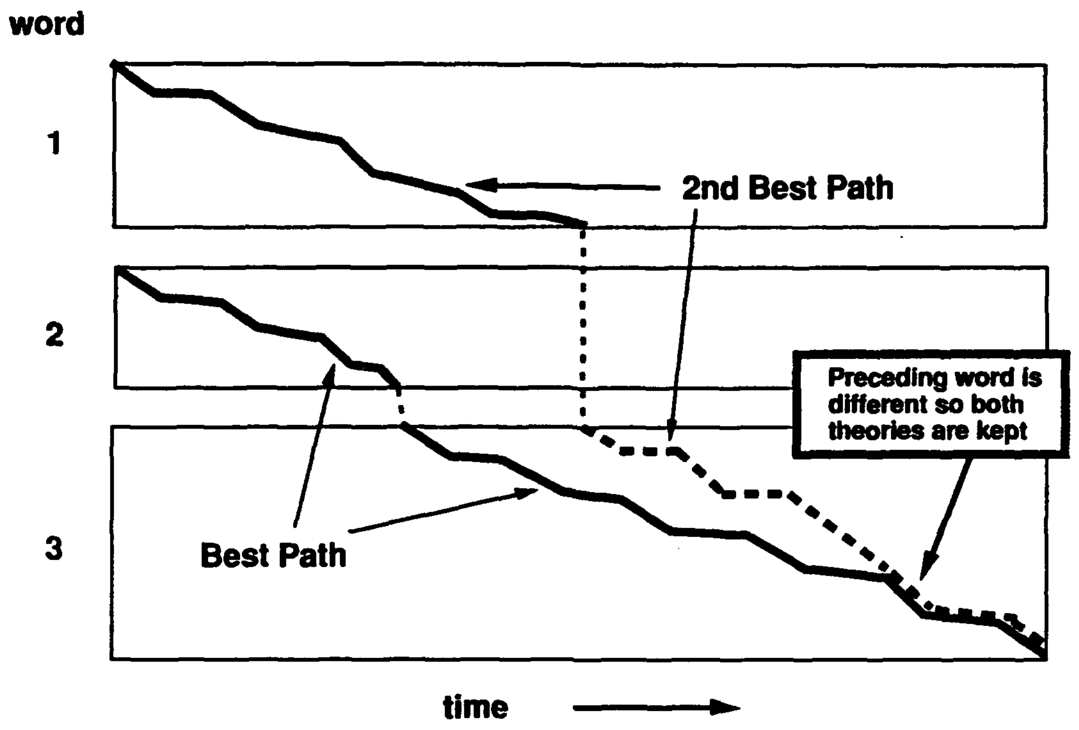


Figure 6.5: *Alternate paths in the Word-Dependent algorithm.* Best path for words 1-3 is preserved along with path for words 2-3.

three algorithms. As can be seen, all three algorithms get the sentence correct on the first choice about 62% of the time. All three cumulative distributions increase substantially with more choices. However, we observe that the Word-Dependent algorithm yields accuracies quite close to that of the Exact Sentence-Dependent algorithm, while the Lattice N-Best is substantially worse. In particular, the sentence error rate at rank 100 (8%) is double that of the Word-Dependent algorithm (4%). Therefore, if we can afford the computation of the Word-Dependent algorithm, it is clearly preferred.

We also observe in Figure 6.6 that the Word-Dependent algorithm is actually better than the Sentence-Dependent algorithm for very high ranks. This is because the score of the correct word sequence fell outside the pruning beamwidth. However, in the Word-Dependent algorithm, each hypothesis gets the benefit of the best theory two words back. Therefore, the correct answer was preserved in the traceback. This is another advantage that both of the approximate algorithms have over the Sentence-Dependent algorithm.

While the new algorithm presented here is quite efficient, it still increases the computation needed by a significant factor over the 1-Best search. Therefore, we developed a technique called the Forward-Backward Search [5]. The algorithm uses a forward Viterbi

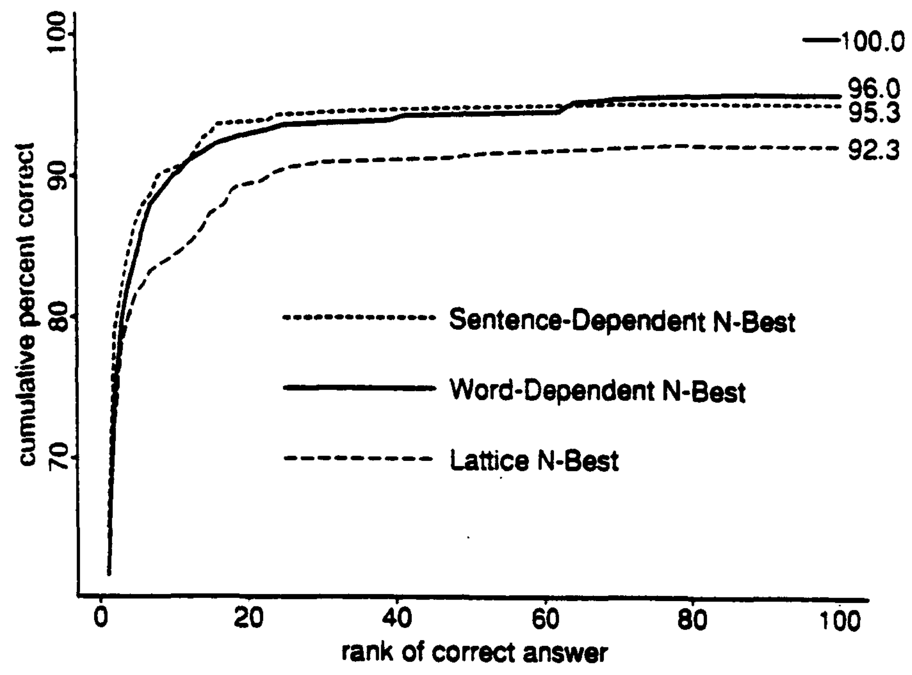


Figure 6.6: Comparison of the Rank of the Correct Sentence for the Sentence-Dependent, Word-Dependent, and Lattice N-Best Algorithms.

search to establish the best score for any path ending with each particular word at each time. This is followed by a backwards version of the Word-Dependent N-Best algorithm, in which only those words that scored well in the forward pass are considered. The result is that the computation of the N-Best list is reduced by a factor of 40. These two algorithms are used in an implementation of a spoken language system that can operate in real time on a commercially available workstation.

6.2.3 Suboptimality of Viterbi Scoring

There have been several other algorithms proposed for finding multiple sentence hypotheses. One algorithm ([95, 62]) is quite similar—with some important implementational differences—to the Lattice N-Best algorithm. This algorithm, as described above suffers severely from the problem that it only considers alternate sequences that diverge from the exact boundaries found for the higher scoring paths. The second algorithm, proposed in [91] is the Tree-Trellis algorithm. This algorithm starts with the same forward-pass as used in the Forward-Backward Search. However, it uses a stack search in the backwards direction to find the N-Best word sequences. It must rescore each hypothesis explicitly in the backwards pass in order to guarantee finding the best sentences. Thus the computation needed is proportional to N, which may be a problem for large N. (The Word-Dependent algorithm requires computation that is proportional to the number of local hypotheses only, which is much smaller than N.)

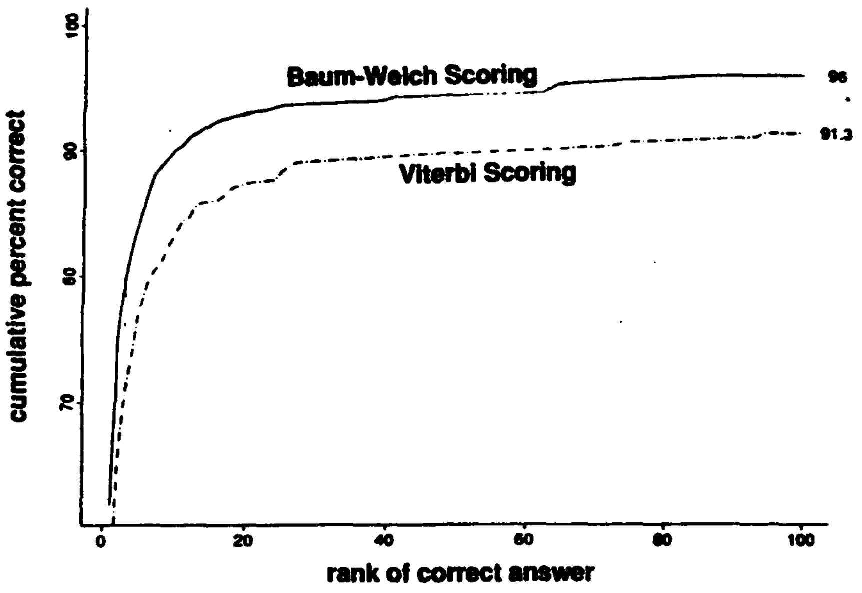


Figure 6.7: Word-Dependent Algorithm using Viterbi scoring vs forward-likelihood scoring. At a rank of 100, Viterbi scoring misses twice as many sentences.

Both of the algorithms mentioned above are based on Viterbi scoring. That is, they find the word sequence corresponding to the most likely single state sequence. However, we should sum the probability over all possible state sequences that correspond to a word sequence. Figure 6.7 shows a cumulative distribution of the rank of the correct answer for the two types of scoring within the Word-Dependent N-Best algorithm. At all ranks, there is a 5% increase in the likelihood of finding the correct sentence when we sum the probability over state sequences. While this difference may not be important for rank 1, where the sentence error rate is 38%, it represents a factor of two in the error rate at a rank of 100. Thus, we feel that it is essential, within an N-Best search, to use the total likelihood score for the word-sequence.

6.2.4 Summary

In summary, we have considered several approximations to the exact Sentence-Dependent N-Best algorithm, and evaluated them thoroughly. We show that an approximation that only separates theories when the previous words are different allows a significant reduction in computation, makes the algorithm scalable to long sentences and less susceptible to pruning errors, and does not increase the search errors measurably. In contrast, the Lattice N-Best algorithm, which is still less expensive, appears to miss twice as many sentences within the N-Best choices.

6.3 New Uses for the N-Best Sentence Hypotheses Within the BYBLOS Speech Recognition System

Since developing the N-Best Paradigm for the integration of speech with NL, we have found many other uses for it. Below, we describe four different ways in which we use the N-Best paradigm within the BYBLOS system. The most obvious use is for the efficient integration of speech recognition with a linguistic natural language understanding module. However, we have extended this principle to several other acoustic knowledge sources. We also describe a simple and efficient means for investigating and incorporating arbitrary new knowledge sources. The N-Best hypotheses are used to provide close alternatives for discriminative training. Finally, we have developed a simple technique that allows us to optimize several weights and parameters within a system in a way that directly minimizes word error rate. These techniques were invaluable in our experimental work on the ATIS domain, in which we would like to use powerful knowledge sources such as trigram statistical models of language.

While most of these uses are not search strategies, they are all related in that we use the N-Best process to provide a reduced set of choices that require special attention. In the case of a search strategy we can apply additional KSs to make the final choice, while in the case of discriminative training, we modify the parameters of our model specifically to reject those wrong answers that our system is likely to choose. In the remaining sections we discuss each of the different uses of the N-Best paradigm within the BYBLOS system.

6.3.1 Efficient Search Strategies

The initial use of the N-Best paradigm to integrate speech recognition and natural language within a spoken language system is an example of a search strategy optimization. That is, we reduce the number of choices using an inexpensive acoustic model and statistical language model before applying the more expensive linguistic understanding model to verify that a sentence hypothesis is meaningful. Given that a linguistic natural language is typically not much more powerful than a simple statistical n-gram language model, yet is much more complex, we find it more cost effective to use the statistical n-gram model in an N-Best algorithm to generate all the likely sentence hypotheses, and then apply the linguistic language model to the resulting alternatives. That is, we use the N-Best algorithm as a sentence-level "fast match". The question, then, is whether the accuracy decreases when using this strategy, since it is not admissible. However we showed in [32] that, given powerful acoustic and language models, the correct sentence is found within the set of proposed sentence hypotheses a very high percentage of the time. In subsequent work, we found that even in those cases where the correct answer was absent, there was almost always a higher scoring answer that was acceptable to the NL module. This means that a more tightly coupled search could not have resulted in lower error rate.

We have extended the basic idea outlined above to several other expensive knowledge sources that we use in our continuous speech recognition system. Some examples of these are: higher-order statistical language models, cross-word coarticulation models, semi-continuous tied-mixture models, and whole-segment acoustic models. Each of these models can result in higher recognition accuracy, but at a significant increase in computational cost and complexity. Instead, we are able to get the benefits of these more detailed models at a fraction of the cost.

High-Order Language Models

Time-synchronous speech recognition using a bigram statistical language is quite efficient. However, if sufficient text training is available, we can derive a more powerful language model of sequences of three or more words or word classes. But the computation needed to use a trigram language model in a time-synchronous search is proportional to the square of the number of words or classes, making it impractical. Instead, we use a bigram model (of word classes) when we find the N-Best hypotheses, and then compute, for each text hypothesis, the language model probability using a 3-gram or 4-gram model. The computation and substitution of the higher-order language model score for a particular known sequence of words and subsequent recombination with acoustic score requires almost no computation, and can thus be accomplished with no perceivable delay. Again, there is the risk that we will make a search error by using the weaker language model to filter out most of the choices. For example, the perplexity of the higher-order language model might be 30% less than the perplexity of the bigram model. In principle, this means that the average number of sentences predicted by the bigram model may be a few orders of magnitude more than predicted by the trigram model, implying that we would need to have N-Best lists that include thousands or millions of sentences to insure having the optimal answer. However, this analysis ignores the fact that the added linguistic information is smaller when used together with an acoustic model, as in speech recognition. (In other words, the increase in mutual information is small.) Thus, we find empirically, that as long as the utterances are not too long, the highest scoring sentence using a trigram model is never past rank 100 when using a bigram language model. (We divide very long utterances into sections and find the N-Best sentences for each section.)

This is not to say that we can always use any weak language model to find the N-Best sentences. For example, if we use a model that simply allows any sequence of words, we find that the correct answer is not in a list of 100 sentence alternatives a sufficiently high percentage of the time. This is also true when we are dealing with severely degraded speech, where the acoustic information is weaker. Thus, we must determine for each case which language model should be used for finding the N-Best sentences. Luckily, it is easy to determine during development when we have made a search error with this paradigm. All we have to do is to (artificially) include the correct answer among the choices. If we find that this improves the accuracy, then this tells us that we have made a search error by using the weaker language model first.

Cross-Word Coarticulation Models

One of the improvements that has been demonstrated in recent years is the use of context-dependent models that span the boundaries between words. It has been shown that modeling acoustic coarticulation between words (cross-word models) reduces the recognition error rate by a significant factor—typically 30%. While the principle is the same as modeling coarticulation within words, there is a large difference in the cost. Since the phonetic pronunciations of a word are essentially a linear sequence with few alternatives, we can replace each phoneme model within a word with the appropriate model that depends on the preceding and following phoneme without significantly changing the topology of the model of the word. However, if we include models that depend on neighboring words, it interacts with the choices in the grammar. In order to use cross-word models, we must compile a triphone network grammar that has a large number of initial and final acoustic models for each word and correctly incorporates the constraints depending on the preceding and following words. This grammar and the resulting computation is quite large.

Again, we can use the simpler models (that model within-word coarticulation only) to compute the N-Best hypotheses and then rescore each hypothesis using cross-word models. To do this we must compile a "grammar" for each sentence hypothesis that incorporates the cross-word models for that sentence, and then use this grammar to compute the probability of the acoustics (i.e. compute the forward probability) independently. Since each hypothesis is a known linear sequence of words, the resulting grammar is quite simple and requires very little computation to evaluate the correct score.

Semi-Continuous Density HMM Models

HMMs that use semi-continuous [45] or tied-mixture densities [16] have been shown to result in slightly less error than the corresponding discrete density HMMs. Semi-continuous density models are like discrete models, except that rather than representing the input with the index of the closest prototype vector, we preserve several (e.g. the 5) most likely regions of the input space, along with their probabilities. This amounts to a smoothing of the probability density at a state between regions in the input space. However, the computation for evaluating the spectral observation probabilities increases several fold, (usually by a factor of 5 to 10), resulting in a total increase in computation by a factor of 2 or more. Consequently, we have considered several alternatives to using the semi-continuous densities in all stages. First, we found that if we just use semi-continuous densities during the training, but use the discrete (top 1) model during recognition, we obtain most of the gain attributed to semi-continuous densities. In order to gain the remaining error reduction, we include the top 5 VQ bins during the rescoring pass. Although the semi-continuous density computation in the rescoring pass is also more expensive, this cost can be minimized because the different sentence hypotheses in the N alternatives typically only span a small fraction of the phonetic models. Therefore, when we rescore each of these alternatives, the extra cost for using semi-continuous densities can be made small.

6.3.2 Investigating and Using New KSs

Usually in order to evaluate the utility of a new KS, it must be implemented in a reasonably efficient recognition search. But this is not always easy. For example, a segmental model of speech (that models whole segments as a single unit rather than as individual frames like an HMM) requires a tremendous amount of computation, making the research with this new model difficult. A model of sentential stress may require relative measurements of pitch, loudness, and phoneme duration over long intervals in a sentence, and may also require relating these measurements to a syntactic grammar. Often, the new KS is not expected to perform the whole recognition problem by itself. Very often the issue of what information the KS provides and how to use it in an integrated system are conflated.

The first question we want to ask about a new KS is not what recognition accuracy results from using this KS by itself or how it interacts in a search strategy, but how much information it adds to the KSs that are already being used. Only after we determine that it is useful do we want to consider the most efficient way to use it. The problem of evaluating a new KS is made simple using the N-Best paradigm. The only requirement for evaluating (or using) a new KS is that there be a way to produce a score for a given sentence hypothesis. The scoring mechanism does not need to be able to operate in a left-to-right manner or on partial sentences. First, we start with a long list of the N-Best hypotheses for each of the utterances in a development test set. (If the correct answer is not included in the list, we can artificially include it in order that the new KS will have a chance to improve its score sufficiently to raise it above all the other hypotheses.) Then we compute the score for each hypothesis using the new KS. We find the optimal linear weights for combining the score of the new KS with those of the previous KSs using the technique described in subsection 5. The decrease in word or sentence error rate—if any—tells us how useful this new KS is in the context of the total system.

If the new KS is found to be useful, then the same method provides a simple and efficient way to integrate it into the system. Of course, if this new KS is powerful enough, and not expensive, we may want to consider ways to incorporate it at an earlier stage.

Segmental Models

We have used this paradigm to utilize segmental models for speech recognition. Two examples of such models are the Stochastic Segment Model [70] and the Segmental Neural Network (SNN) [7]. The advantage of a segmental model of speech is that it can take into account dependency between frames within a phoneme. However, the computation required for recognition is greatly increased. Even though we can use a dynamic programming update to compute different path scores, we must explicitly consider different segment durations for each possible ending time for a segment, resulting in an order of magnitude more computation than required for typical frame-based HMM models. The computational and search issues related to segmental models are avoided by using the N-Best paradigm, and we can measure directly the improvement when added to our best HMM system.

One byproduct of rescoring the N-Best alternatives with the HMM is the best phoneme sequence along with the best corresponding segmentation for each sentence hypothesis. Each of these hypotheses can be rescored quickly using the segment model, since there is no ambiguity as to the segmentation. If we allow the segmentation to be different for the segmental model than for the HMM, we can speed up the search tremendously by constraining the segmentation to be close to that of the HMM.

This approach has made it possible to consider using models that would otherwise be too expensive. While the recognition accuracy with the two new segmental models was not—by itself—superior to that of the HMM, we found that when we combined the new scores with the previous scores, we were able to obtain better performance than with either model alone. Specifically, in the case of the SNN, we performed experiments on the DARPA 1000-Word Resource Management Corpus. We were able to reduce the word error rate on the October '89 speaker-independent test set from 4.1% to 3.0% by using the combined HMM and SNN scores [7]. This represents a substantially lower error rate than previously reported on this corpus.

6.3.3 Discriminative Training

Several methods for discriminative training have been proposed. These methods generally need a mechanism for creating errors and near misses in order that the correct answers can be made more likely, while the errors and all of the near misses can be made less likely. Usually, ad hoc techniques are used to derive a list of near misses for each word independently. Instead, we compute the N-Best sentence hypotheses, and rescore them using all of the KSs in the system. This means that any errors that show up in the final list are real errors that the system is likely to make, and we should attempt to remove them by discriminative training. Conversely, if a particular error is unlikely, e.g. because the grammar makes it unlikely, we won't observe it and we don't need to spend modeling effort to avoid that error. We have reported on using this method for MMI training [30] of the codebook weights in a multiple codebook discrete density HMM system.

This method has been most successful within the context of the Segmental Neural Networks(SNN). In our initial implementation of SNN we trained the networks on speech that was phonetically segmented using trained HMM models. Each segmented phoneme served as a positive example of its own model and a negative example for all of the other models in the usual manner of neural network training. There are two problems with the approach. First, the SNN is required to model all of the distinctions between each phoneme and all of its neighbors. But this job is already being done quite well by the HMM. Second, the SNN is only trained on correct segmentations. But we know that in practice, almost all recognition errors also result in a different phonetic segmentation than that of the correct answer. To be effective as a discriminator, the SNN must learn to reject these incorrect segmentations. However, rather than training the segmental neural networks on all possible segmentations and labelings, we would like it to concentrate all of its modeling capability on eliminating those few errors made by the HMM.

First, we use the correct labeling and segmentation as provided by the HMM models as examples of the correct models. Then, we use the N-Best list (rescored with detailed HMMs and language models) as examples of likely misrecognitions by the HMM. We determine, for each incorrect hypothesis, which phonemes differed from the correct answer in labeling and/or segmentation. The difference usually only consists of a few phonemes in each hypothesis. These incorrect phonemes are used as negative examples for training the SNN. The result of this training method is that the models are better able to complement the HMM models as an additional KS.

6.3.4 Optimization Of System Parameters

In addition to the large number of acoustic and language model parameters in a recognition system, there are several system parameters that must be tuned for optimal performance. Many of these cannot be estimated directly using the same techniques (e.g. maximum likelihood). Some examples of these parameters are: word and phoneme insertion penalties, the grammar weight, the codebook weights, and the weights of alternate acoustic models. The word insertion penalty is an additional probability that we multiply by for each transition to a new word (in addition to the grammar probability). This is used to control the balance between insertions and deletions. The language model weight is an exponent on each grammar probability that allows us to obtain the best balance between the acoustic model scores and the language model scores. Finally, each different acoustic model is weighted by exponentiating the probabilities according to their relative power.

Clearly we would like to set these parameters to optimize recognition accuracy directly. However, maximum likelihood estimation techniques cannot be used to estimate these exponent parameters. Therefore, we typically run several recognition experiments—each requiring a few hours—to try to find the best system parameters. However, this tuning often requires extensive experience and too many experiments.

The total probability of a sentence hypothesis can be expressed as the product of the exponentiated probabilities of each KS.

$$\begin{aligned}\text{Utt-score} = {} & \text{HMMScore}^{\alpha} * \\ & \text{GrammarScore}^{\beta} * \\ & \text{WordPenalty}^{\#words} * \\ & \text{PhonePenalty}^{\#phones}\end{aligned}$$

The unknown values are the exponents α and β, and the WordPenalty, and PhonePenalty. If we take the log, we have

$$\begin{aligned}\log \text{Utt-score} = {} & \log \text{HMMScore} * \alpha + \\ & \log \text{GrammarScore} * \beta + \\ & \#words * \log \text{WordPenalty} + \\ & \#phones * \log \text{PhonePenalty}\end{aligned}$$

Now, the unknown values on the right are just linear weights for the KSs on the left. Admittedly the number of words and phones are simple KSs, but we find that including these terms significantly improves recognition accuracy. We need to find the four values that minimize the error rate. While minimizing error rate directly for continuous speech is usually difficult, it becomes easy if we change the problem to one of minimizing the error rate when choosing among the N-Best alternatives for an utterance. First, we find the N-Best hypotheses for all of the utterances in a development test set. The rescoring step provides the log probabilities for each hypothesis for each KS separately. We use a gradient search to find the set of weights that, averaged over the development set, brings to the top the answer with the smallest number of errors. To evaluate a particular set of weights we compute the total weighted log score for each hypothesis (the dot product of the weights and scores), and then find the hypothesis with the maximum total score for each utterance. We measure the word error rate for this top choice for each utterance in the set in the usual way. The total word error rate over the set is our evaluation function for this set of weights. The computation needed to evaluate a set of weights for 100 hypotheses for 300 test utterances can be measured in milliseconds. Therefore we can consider several thousand weight vectors in a few seconds in our search for the set of weights that minimizes word error rate on the development set. As long as the development test set contains enough utterances—say 300—we find that the weights found are also good for new test sets.

6.3.5 Summary

We have described several new uses for the N-Best paradigm. As originally intended, it is useful for decreasing computation when using several different knowledge sources for speech recognition. But it has also been shown to be quite useful for discriminative training of several different types, and for directly optimizing final system performance, which previously required an exhausting number of experiments.

6.4 Integration with Natural Language

We experimented with several conditions to optimize the connection of BYBLOS with DELPHI. The basic interface between speech and natural language in HARC is the N-best list. In our initial implementation of this integration strategy, we simply allowed the natural language component to search arbitrarily far down the N-best list until it either found a hypothesis that produced a database retrieval or reached the end of the N-best list. Over the course of the rest of this project, we explored the nature of this connection in more detail. The parameters we varied were:

- the depth of the search that NL performed on the N-best output of speech
- the processing strategy used by NL on the speech output

In our examination of the data from the February, 1991 cross-site evaluation, we had noticed that while it was beneficial for NL to look beyond the first hypothesis in an N-best list, the answers obtained by NL from speech output tended to degrade the further down in the N-best list they were obtained. Subsequently, we performed a number of experiments to determine the break-even point for NL search. We used an N of 1, 5, 10, and 20 in our experiments.

During our most recent development work, we utilized the fall-back strategies for NL text processing described in 4.3. In applying these fall-back strategies to speech output, we examined the trade-off between processing speech output with a more restrictive scheme, and thereby potentially discarding meaningful utterances vs. processing speech output with a more forgiving strategy, and thereby potentially allowing in meaningless or misleading utterances. We experimented with three processing strategies:

- fallback processing turned off
- fallback processing turned on
- a combined strategy, in which an initial pass with made with fallback processing turned off. If no hypothesis produced a database retrieval, a second pass was made, with the fallback strategy engaged.

We show the results of one such experiment, utilizing the October, 1991 dry-run corpus as development test in Table 6.1.

The results of our experiments indicated that an N of 5 was optimal, and that the two-pass processing strategy was slightly better than either of the others. This was the configuration we used on the February, 1992 cross-site evaluation data.

We repeat here our results on the February, 1992 cross-site evaluation data, originally presented in Table 4.3.

Condition	N	WE
Text	(1)	47.9
Fallback on	1	64.6
"	5	58.0
"	20	60.1
Fallback off	1	64.2
"	5	56.9
"	20	59.0
Two Pass	5	56.6

Table 6.1: SLS weighted error (WE) on the October '91 dry-run test set with varying N-best list length (N).

	%T	%F	%NA	%WE
syn only	68.9	11.5	19.7	42.6
frame only	73.8	13.0	13.2	39.2
both(official)	71.8	15.4	12.8	43.7
both(adjusted)	71.9	15.1	13.0	43.2

Table 6.2: SLS Results, February, 1992

Chapter 7

Real-Time Speech Recognition

One of the major goals of this contract has been the development of real-time spoken language systems. It has long been believed that the computation required for speech recognition and understanding is beyond the capability of general purpose computers. The search space for speech recognition alone is large. In addition, if we include a natural language model, the search space for speech becomes much more complex, and the natural language computation can be many orders of magnitude more than required for text processing.

The traditional approach to solving this problem has been the development of special purpose hardware, or the use of large parallel processor systems. Unfortunately, both of these approaches present new problems of their own. In addition, the speed advantages from these approaches may be no more than a factor of 10. Frequently, before these hardware efforts finish, the speed of the general purpose machines has increased to the point where there would be no advantage for the special purpose hardware.

We have come to believe that the most profitable approach to the problem is to think about algorithms that give the same accuracy with a small fraction of the computation. We believe this because we have been successful in developing many such approaches which, taken together, save several orders of magnitude in computation with no loss in accuracy. Consequently, our goal has been to use readily available workstations for the computation engines, and obtain increased speed from a combination of new search algorithms and the natural evolution in hardware speed.

In this chapter, we describe some of the algorithms that we have developed to enable real-time spoken language understanding. In particular, these are the Forward-Backward Search Algorithm and an efficient algorithm for decoding with statistical n-gram grammars.

We also demonstrate convincingly that, when searching for the most likely sequence of words, it is better to add the probabilities from different state sequences for a given word sequence (the Forward Probability Search) rather than simply to find the word sequence corresponding to the most likely state sequence (the Viterbi algorithm). We also describe the hardware that we used, and review some of the demonstrations that we have given along the way.

7.1 Forward-Backward Search Algorithm

Despite the advances in the speed of the computation of the N-Best sentences, it still requires more computation than we would like to spend. We have developed a general technique that greatly speeds up expensive time-synchronous beam searches in speech recognition. The algorithm is called the Forward-Backward Search and is mathematically related to the Baum-Welch forward-backward training algorithm. It uses a simplified forward pass followed by a detailed backward search. The information stored in the forward pass is used to decrease the computation in the backward pass by a large factor. We have observed an increase in speed of a factor of 40 with no increase in search errors.

7.1.1 Introduction

As speech recognition algorithms advance in complexity, so does the amount of computing power they require. At some point we start looking for ways to make them run faster, both because we want to run experiments in a reasonable amount of time and also because we want to incorporate these new algorithms into real-time speech understanding systems. The Forward-Backward Search (FBS) can be used to obtain increases in speed of up to 40 in time-synchronous algorithms by using information obtained in a fast, simplified algorithm. It is so called because of its similarities to the Baum-Welch forward-backward training algorithm.

First, we briefly describes how fast-match algorithms work and identifies some trade-off problems in their implementation. Next, we will describe the Forward-Backward Search and subsection 4 describes some considerations in its implementation. Finally, we describe how we use the FBS in our real-time speech understanding system.

7.1.2 Fast-Matc ı Algorithms

A time-synchronoı s HMM CSR system works by taking each frame of an utterance and simultaneously matching that frame to a set of HMMs. In order to limit the amount of work performed by the recognizer, it is customary to apply a *pruning beamwidth* or *threshold* to the scores generated by the HMMs, excluding from the match any HMM whose score falls below the threshold. HMMs which are included in the match are described as "activated" and those that are excluded are "deactivated".

In addition to the mechanism for deactivating HMMs provided by the pruning beamwidth, we need a method for activating HMMs so that, as the utterance progresses, the HMMs corresponding to the new portions of the speech may be brought into the match. This is usually done with a grammar. The grammar takes the final state score of an active HMM and sets the scores of each of the possible following HMM's initial state to the final score multiplied by a grammar score. If the score of this new HMM is above the pruning threshold, the HMM is made active and henceforth included in the match until deactivated.

Since the amount of work performed by the recognizer is dependent upon the number of active HMMs and since this strategy pays no attention to the utterance beyond the present frame, many models are made active, only to be deactivated a number of frames later, having done no useful work. In addition, even if the word ends with a good score, there may not be any path to the end of the utterance that is both grammatically and acoustically correct. Since it may take many frames of the utterance for the score of such an activated HMM to fall below the pruning threshold, most of the active models in a CSR system may be in this situation and consequently most of the work performed in matching these HMMs to the utterance will ultimately be wasted. A fast-match algorithm could be used to reduce the number of these needlessly activated models by looking at the next few frames of the utterance, but it would be better if we had some way of predicting whether the activation of a particular model at a particular point in an utterance is consistent with the rest of the utterance.

The usual method of implementing a fast-match algorithm[11, 38] in a CSR system is to use two recognition systems in tandem. The first system acts as a filter to prune out words which have no chance of being recognized by the second, more detailed, system. In a time-synchronous CSR system, the fast match recognizer examines the speech signal for a section of speech after the time which the second system has reached and performs a coarse analysis of the speech signal. The first system analyzes this section of speech and compares it to models of the initial portions of all the words in the vocabulary and compiles a list of words which might start during the "lookahead" portion of speech. The second, frame-synchronous, system operates in the usual manner (e.g. the Viterbi decoding algorithm) except that it does not consider starting words not in the list compiled by the fast-match algorithm.

Fast match algorithms suffer from the disadvantage that they have to be *tuned*—i.e. there are arbitrary parameters of the algorithms which have to be adjusted in order to achieve a trade-off between the selectivity (which words are included in the list) against the chance of a search error (i.e. a situation where a word that is actually starting is not included in the list).

Since a fast-match algorithm works by examining a lookahead portion of an utterance to determine whether a word is likely to begin, there is a trade-off in the length of the lookahead region: in order to be able to omit as many words as possible from the search, it is desirable that the fast-match algorithm look as far ahead as possible, but the more the fast-match looks ahead, the more the detailed match is delayed. This is because when the fast-match reaches the end of the utterance, the detailed match must be delayed by at least the length of the lookahead region. Thus, typically, no attempt is made to confirm that the speech beyond is consistent with the hypothesized word.

Also, as the fast-match algorithm has to work at high speed, the length of this lookahead region must be quite small. This leads to a problem. For example, if a fast-match algorithm chose a word on the basis of the first two phonemes, there would be no way of distinguishing between "ADD" and "ADDITIONAL". If the utterance was "... ADD AN AREA ...", the inclusion of "ADDITIONAL" would be wasteful, since it would result in a bad match for the phonemes beyond the first two. Although a fast-match algorithm may reduce the amount of work of the second system, its effectiveness is hampered both by its near-sighted nature and by the fact that the acoustic match used in the fast-match criterion is inferior to that used by the second system. Thus a fast-match system must either be cautious or make mistakes.

The FBS avoids these trade-off problems while also achieving increases in speeds by a factor of at least 40.

7.1.3 The Forward-Backward Search (FBS)

As its name suggests, the FBS takes place in two phases. The first phase performs a fast time-synchronous search of the utterance in the forward direction. The backward pass performs a more expensive search, processing the utterance in the reverse direction and using information gathered by the forward pass. It is used when implementing a complex algorithm, such as the N-best[85, 32] algorithm which stands little chance of being performed in real-time by itself.

The FBS is an algorithm which does this and can be used when one of the following cases applies.

- A simplified form of the algorithm is available. For example, the N-best search is an expensive time-synchronous algorithm which produces the N most likely sentence hypotheses for a given utterance. A simple version of this is the Viterbi decoding algorithm which only produces one hypothesis for an utterance, but which takes far less time computationally.
- A simplified acoustic model is available. In the BYBLOS system, we use 3-state triphone HMMs. Other CSR systems might use more complex acoustic models. A simplification of this could be to use 1-state models.
- A simplified language model is used. We have performed experiments using both very large statistical/finite-state grammars and rule-based grammars. By using the FBS with a simplified grammar, we have been able to run experiments which would have been impossible due to the large amount of grammar work involved.

In such cases it is possible to use information generated by the simplified recognition strategy to greatly reduce the amount of time taken in a second pass of the algorithm. For a real-time system, the implication of this is that the simplified forward pass can be performed in real-time, and the backward pass can be performed in a very short time after the utterance has finished. The delay involved in waiting for the backward pass is about the same as the delay needed to implement a fast-match lookahead. For example, the N-best algorithm is computationally expensive and takes more than 20 times real-time to compute for the 1000 word resource management task. We can however perform a simple Viterbi time-synchronous beam search in real-time and use information gathered from this to speed up the N-best algorithm by about a factor of 40 to a point where it finishes with only a short delay.

The key to the FBS is that while the forward search is performed, the scores of the final state of each active HMM which represents a word ending are recorded at each frame of the utterance together with a record of which HMMs were active. We will denote the set of words which are active at time t of the utterance as Ω^t and the scores of the final states of each word ω in Ω^t as

$$\alpha(\omega, t)$$

We will also need the maximum score of all sentence hypotheses, i.e. the score of the optimal path determined by the forward pass. We will call this α^T.

After the simplified forward pass has been completed, the second, expensive algorithm is performed *in reverse*—that is, the second pass starts by taking the final frame of the utterance and works its way back, matching frames earlier in the utterance until it reaches the start of the utterance. At each frame, t, of the utterance, the exit score of each active HMM (which corresponds to the initial HMM state, because the HMMs are operating in reverse) is worked back though the grammar to give an input score for the each HMM linked to it. We call such a HMM ω' and the score which the grammar has computed with the aid of the previous HMM is called

$$\beta(\omega', t)$$

If ω' is not already active, we can use the FBS to decide whether it is worth making it active. The first thing to note is that if ω' is not a member of Ω^t, we can conclude, solely on the basis of the portion of the utterance between the start and t, that all paths from ω' to the start of the utterance resulted in such low scores that they would be pruned from the recognition. Therefore, if $\omega' \notin \Omega^t$, we do not make the HMM active.

This in itself results in a great saving since it drastically reduces the number of HMMs active. The algorithm has effectively "looked into the future" to decide if the HMM has a chance to feature prominently in the final recognition result. However, it is also possible to check whether the scores from the beginning of the utterance to t and from t to the end of the utterance together imply that all paths passing through ω' at t will be pruned at some time.

The maximum score for all sentence hypotheses passing though ω' at time t is

$$\alpha(\omega', t)\beta(\omega', t)$$

and the maximum score for all sentence hypotheses is α^T. Thus, if the quantity[1]

$$\frac{\alpha(\omega', t)\beta(\omega', t)}{\alpha^T}$$

falls below the pruning threshold, we can conclude that all paths that pass though the final state of ω' at t will eventually be pruned. If this is the case, again it is useless to activate ω' at this point in the utterance.

Figure 7.1 describes how the FBS works. It depicts a situation which could arise during the backward pass of the FBS. The paths coming from the top left hand corner represent the terminal scores collected from the forward pass and those from the top right represent the scores in the backward pass which are offered to the terminal states of models **a**, **b**, **c** and **d**. The distance from the top of the figure represents the logarithm of the score, so the product of two numbers is represented by the sum of the distances from the top of the figure. For this value of t, path **d** can immediately be eliminated since, although it is currently the highest scoring path in the backward pass, its exclusion from

$$\Omega^t = \{\mathrm{e}', \mathrm{b}', \mathrm{a}', \mathrm{c}'\}$$

[1]This quantity is very similar to the expression for the likelihood calculated as part of the Baum-Welch forward-backward training algorithm (in this case we are calculating the *maximum* likelihood as opposed to the *total* likelihood used in the training algorithm) and is the reason behind the naming of the FBS.

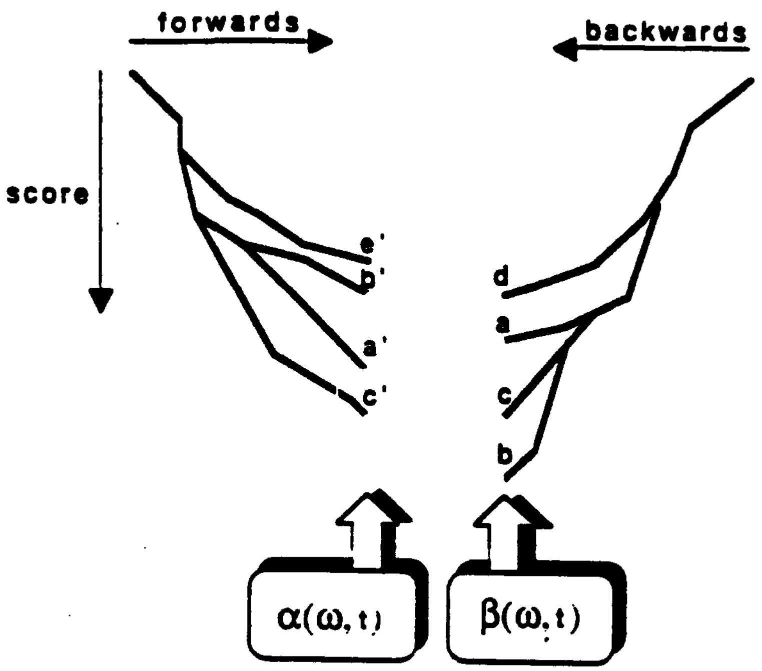


Figure 7.1: Forward-Backward Search. Forward and backward scores for the same state and time are added to predict final score for each theory extension.

indicates that it scores so badly in the portion of the utterance from t down to the beginning that it will be pruned. The elimination of the other three paths, **a**, **b** and **c** depends on the pruning beamwidth. Since

$$\alpha(\omega, t)\beta(\omega, t)$$

is the sum of the "depths" of the two paths at t, path **c** is most likely to be eliminated, followed by **b** and finally **a**.

A consideration in using the FBS for CSR systems which use HMM to model phonemes (as opposed to whole words) is that the recording and checking of HMM scores can be restricted to a subset of the HMMs. For instance, in CSR systems which use phoneme-based HMMs which are concatenated to represent whole words it is more efficient to only consider the phoneme HMM which represent the ends of the words. Phoneme-based HMM CSR systems often join up strings of phonemes to represent words. Since the HMMs representing the end of the word will only have a significant score if the whole word matches, few word-end HMMs will be active. In the backward pass many of the word-end phonemes could become active, since the recognition is proceeding in reverse, but the application of the FBS limits these prevents this. Since the word-ends do not become active in the backward pass no other phoneme model in the word can be activated.

7.1.4 Implementation of the Forward-Backward Search

Perhaps one of the most appealing features of the FBS is the ease with which it can be implemented. The first task in the implementation is to make the second pass operate in reverse. This can be done in two ways: the algorithm can be made to scan the utterance in reverse and the state and grammar transitions can be traced backwards or the utterance itself can be reversed as well as the transitions in the HMMs and grammar and the actual algorithm may remain unchanged. When converting an existing algorithm to use the FBS, the easiest way may be to use the second strategy, but for ease of notation, we will adopt the first.

```
t = T;
while ( t > 0)
  t = t-1
  for each $\omega$ ActiveWords
    UpdateAllStatesWithinWord();
    X = GetSetOfAllBackwardLinkedHmms();
    for each $\omega'$ in X
      $\beta(\omega', t)$ = GetHmmScoreWithGrammar($\omega, \omega'$)
    end
    MaxBeta = Max($\beta(\omega', t)$)
    for each $\omega'$ in X
      if ($\beta(\omega', t) >$ PruningBeamwidth * MaxBeta AND
        $\omega' \in \Omega^t$ AND
        $\alpha(\omega', t) * \beta(\omega', t)/\alpha^T >$ PruningBeamwidth)
      begin
          AddHmmToActiveWords ($\omega', \beta(\omega', t)$)
      end
    end
  end
end
```

Figure 7.2: Pseudo-code implementation of backward part of Forward-Backward Search.

The forward pass of the FBS proceeds as normal except that additional information is gathered and stored for each time t. We denote the set of phoneme HMMs which are both active at time t and which also correspond to the end of words by Ω^t. We record the set Ω^t, and also for each $\omega \in \Omega^t$, we record $\alpha(\omega, t)$, the score (or likelihood) of the final (or exit) state.

After the backward pass has been modified to run in reverse, a small modification is necessary to complete the implementation. At each t, the backward algorithm must propagate scores from an active HMM (ω) back through the grammar to another HMM (ω') at the same time calculating the HMM/grammar score for ω' for the next frame (actually $t-1$). In a normal beam-search, this would be compared to the pruning threshold in order to limit the number of theories in the search, but in the FBS we apply the additional tests of inclusion in the Ω^t set and of the forward-backward score.

Figure 7.2 shows how a pseudo-code implementation of the backward pass of the FBS might look. The only difference from the original algorithm is the the second and third lines of the "if" statement (indicated by boxed text) which is performed when deciding whether to make a HMM active. Without the FBS, this "if" statement would be something like:

```
if ($\beta(\omega',t)$ > PruningBeamwidth * MaxBeta)
begin
   AddHmmToActiveWords ($\omega',\beta(\omega',t)$)
end
```

7.1.5 Use of Forward-Backward Search in a Real-Time System

BBN's HARC real-time continuous speech recognition systems uses the Word-Dependent N-best[85, 32] algorithm to present an order list of word hypotheses to the DELPHI natural language (NL) processor. The N-best paradigm was chosen as the interface between the two parts of the system both because the sentence accuracy of the optimal (Viterbi) answer from the speech recognition part is not high enough to be usable and also because it is difficult to use a complex natural language decision process at the frame level of the recognition.

The N-best algorithm would, by itself, be impractical to use in a real-time system, because it takes about 20 times real-time to compute. Not only do we use the FBS to speed the N-Best algorithm to run in near real time, but we also use the result of the forward (1-best) pass to provide the natural language system with the most likely answer soon after the end of utterance is detected. Figure 7.3 illustrates how this works.

An MTU A/D converter digitizes the speech and feeds it to a dual C30 board which is housed in a Sun4/330. This board performs analysis and vector quantization on the speech and also detects the start and end points of the utterance. The vector quantized speech is transferred to the main Sun4 via the VME bus. Here, the Sun4 performs the forward pass of the FBS. Soon after the end of the utterance is detected, not only has the

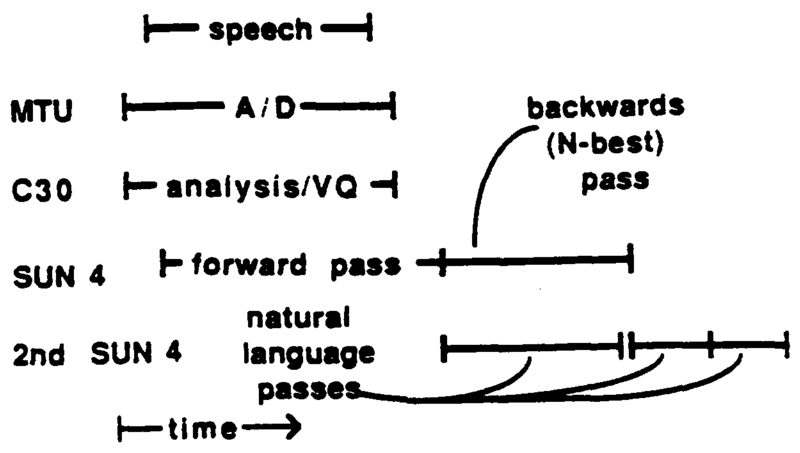


Figure 7.3: Synchronization of Forward-Backward Search with NL processing in the real-time speech understanding system.

forward pass computed the statistics for the forward-backward algorithm, but it has also computed the Viterbi (1-best) transcription of the utterance. This first utterance is then handed to the natural language system which runs on a second Sun4 processor. If this first utterance is accepted by the NL system, then a database enquiry is performed immediately. However, in case the first utterance is not acceptable by the NL system (on syntactic or semantic grounds), the backward-pass of the FBS is performed at the same time as the NL is processing the first answer. Thus, if the NL rejects the most likely answer, the second and subsequent answers are available for the NL to attempt to understand.

7.2 Time-Synchronous Statistical Language Model Search

7.2.1 Basic Search Algorithm

We know that any language model that severely limits what sentences are legal cannot be used in a real SLS because people will almost always violate the constraints of the language model. Thus, a Word-Pair type language model will have a fixed high error rate. The group at IBM has long been an advocate of statistical language models that can reduce the entropy or perplexity of the language while still allowing all possible word sequences with some probability. For most SLS domains where there is not a large amount of training data available, it is most practical to use a statistical model of word classes rather than individual words. We have circulated a so called Class Grammar for the Resource Management Domain [35]. The language model was simply constructed, having only first-order statistics, and not distinguishing the probability of different words within a class. The measured test set perplexity of this language model is about 100. While more powerful "fair" models could be constructed, we felt that this model would predict the difficulty of a somewhat larger task domain. The word error rate is typically twice that of the Word-Pair (WP) grammar. One problem with this type of grammar is that the computation is quite a bit larger than for the WP grammar, since all 1000 words can follow each word (rather than an average of 60 as in the WP grammar).

During our work on statistical grammars in 1987 [78], we developed a technique that would greatly reduce the computational cost for a time-synchronous search with a statistical grammar. Normally, the number of grammar transitions in a bigram statistical grammar is equal to the square of the number of words. (In a trigram grammar it is equal to the cube of the number of words.) However, most of the bigram probabilities are, in fact, estimated from a weighted version of the unigram probabilities, since the bigrams did not occur in the training set for the language model. There are many different methods for combining the bigram and unigram probabilities, including padding the bigram probabilities, a weighted sum based on the number of occurrences of the first word in the bigram, and the backing-off algorithm of Katz. In each of these cases, the grammar structure can be modified to take into account the weighting scheme in such a way that only the bigrams that have actually been observed need to be represented as explicit transitions in the grammar. This greatly reduces the computation need for recognition with a bigram grammar. It is also possible to reduce the computation for a higher-order grammar in an analogous way. This approach is described more fully below.

Figure 7.4 illustrates a fully-connected first-order statistical grammar. If the number of classes is C, then the number of null-arcs connecting the nodes is C^2. However, since the language models are rarely well estimated most of the class pairs are never observed in the training data. Therefore, most of these null-arc transition probabilities are estimated indirectly. Two simple techniques that are commonly used are padding, or interpolating with a lower order model. In padding we assume that we have seen every pair of words or classes once before we start training. Thus we estimate $p(c_2|c_1)$ as

$$p(c_2|c_1) = \frac{N(c_1, c_2) + 1}{N(c_1) + C}$$

In interpolation we average the first-order probability with the zeroth-order probability with a weight that depends on the number of occurrences of the first class.

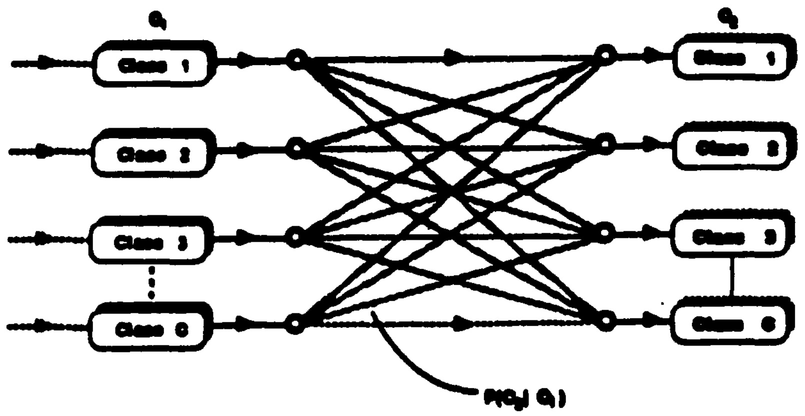


Figure 7.4: Fully Conn.cted First-Order Statistical Grammar. Requires C^2 null arcs.

$$p(c_2|c_1) = \lambda(c_1)\hat{p}(c_2|c1) + [1 - \lambda(c_1)]\hat{p}(c_2)$$

where

$$\hat{p}(c_2|c1) = \frac{N(c_1, c_2)}{N(c_1)}$$

and

$$\hat{p}(c_2) = \frac{N(c_2)}{N(all\ words)}$$

In either case, when the pair of classes has never occurred, the probability can be represented much more simply. For the latter case of interpolated models, when $N(c_1, c_2) = 0$ the expression simplifies to just

$$[1 - \lambda(c_1)]\hat{p}(c_2)$$

The first term, $1 - \lambda(c_1)$, depends only on the first class, while the second term, $\hat{p}(c_2)$, depends only on the second class. We can represent all of these probabilities by adding a zero-order state to the language model. Figure 7.5 illustrates this model. From each class node we have a null transition to the zero-order state with a probability given by the first term. Then, from the zero-order state to each of the following class nodes we have the zero-order probability of that class.

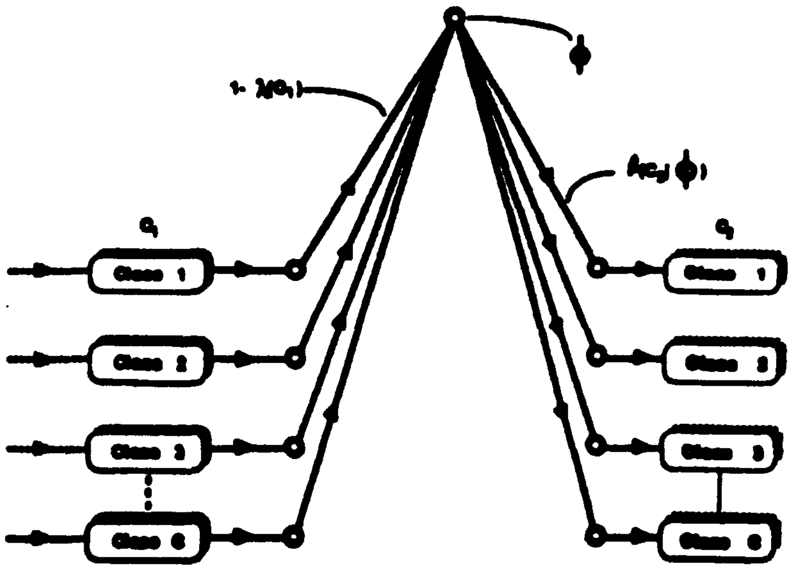


Figure 7.5: Zero-state within first-order statistical grammar. All of the transitions estimated from no data are modeled by transitions to and from the zero-state.

Now that the probabilities for all of the estimated transitions has been taken care of we only need the null transitions that have probabilities estimated from actual occurrences of the pairs of classes, as shown in Figure 7.6. Assuming that, on average there are B different classes that were observed to follow each class, where $B << C$, the total number of transitions is only $C(B+2)$. For the 100-class grammar we find that $B = 14.8$, so we have 1680 transitions instead of 10,000. This savings reduces both the computation and storage associated with using a statistical grammar.

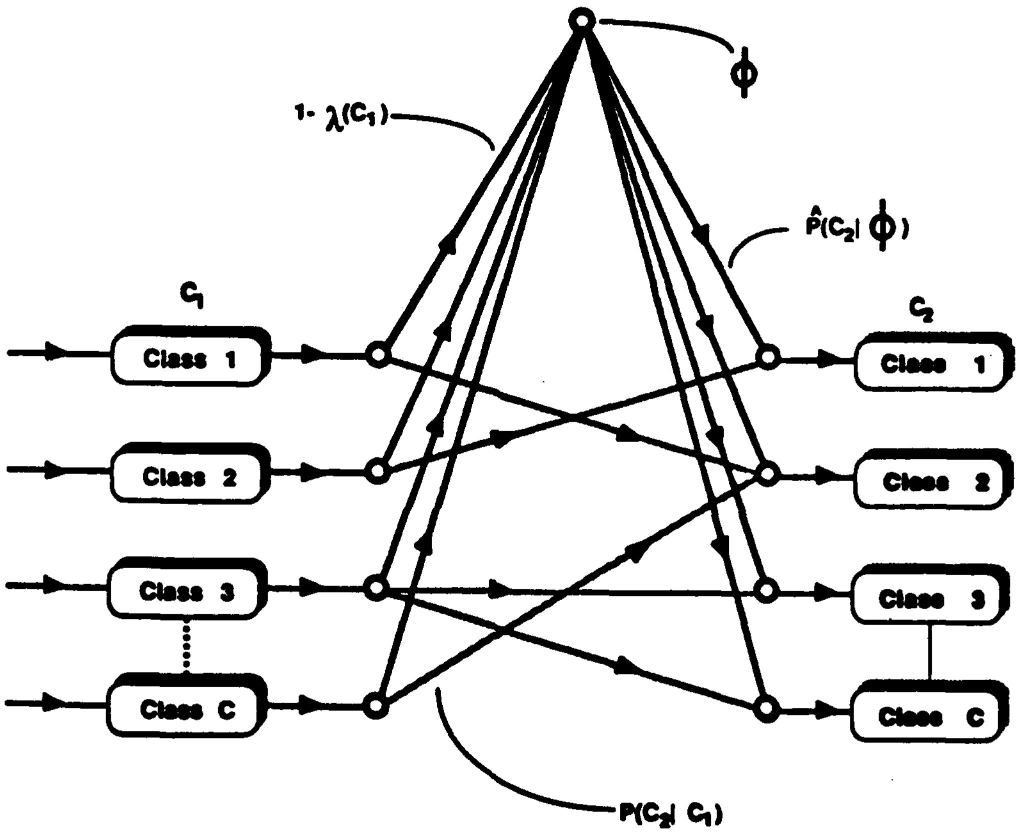


Figure 7.6: Sparsely Connected First-Order Statistical Grammar with zero-state requires much fewer null arcs.

It should be clear that this technique can easily be extended to a higher order language model. The unobserved second-order transitions would be removed and replaced with transitions to a general first-order state for each word or class. From these we then have first-order probabilities to each of the following words or classes. As we increase the order of the language model, the percentage of transitions that are estimated only from lower order occurrences is expected to increase. Thus, the relative savings by using this algorithm will increase.

7.2.2 Time-synchronous Forward Probability Search vs Viterbi

The search algorithm that is most commonly used is the Viterbi algorithm. This algorithm has nice properties in that it can proceed in real time in a time-synchronous manner, is quite amenable to the beam-search pruning algorithm [59], and is also relatively easy to implement on a parallel processor. Another advantage is that it only requires compares and adds (if we use log probabilities). Unfortunately, the Viterbi algorithm finds the most likely sequence of states rather than the most likely sequence of words.

To compute the probability of any particular sequence of words correctly requires that we add the probabilities of all possible state sequences for those words. This can be done with the "forward pass" of the forward-backward training algorithm. The only difference between the Viterbi scoring and the Forward-pass computation is that we add the probabilities of different theories coming to a state rather than taking the maximum.

We presented a search algorithm in 1985 [86] that embodied most of this effect. Basically, within words we add probabilities, while between words we take the maximum. It was not proven at that time how much better, if any, this algorithm was than the simpler Viterbi algorithm, and whether it was as good as the strictly correct algorithm that computes the score of each hypothesis independently.

When we compared these three algorithms under several conditions, we found that there was a consistent advantage for adding the probabilities within the word. For example, when we use the class grammar, we find that the word error rate decreases from 8% to 6%.

To be sure that the time-synchronous forward search gives us the same performance as the ideal forward score is somewhat more complicated. We must guarantee that we have found the highest scoring sentence with the true forward probability score. One way to find this is to use the exact N-Best algorithm [32]. Since the exact N-Best algorithm separates the computation for any two different hypotheses, the scores that result are, in fact, the correct forward probabilities, as long as we set N to a large enough value. A second, much simpler way to verify the time-synchronous algorithm is to see if it ever gets a wrong answer that scores worse than the correct answer. We ran a test in which all incorrect answers were rescored individually using the forward probability. We compared these scores to the forward probability for the correct answer. In no case (out of 300 sentences) did the time-synchronous forward search ever produce a wrong answer that, in fact, scored worse than the correct answer.

The reason that this whole discussion about the Viterbi algorithm is relevant here is that the Viterbi algorithm is faster than the forward search. Therefore, initially, we used the integer Viterbi algorithm in the forward-pass of the Forward-Backward Search. Since the function of the forward-pass is primarily to say which words are likely, it is not essential that we get the best possible answer. The backward N-Best search was then done using the correct algorithm that adds different state-sequence probabilities for the same word sequence. However, more recently we have found that the floating point hardware of the newer workstations is fast enough so that there is no significant speed advantage for the Viterbi search at any point. Therefore, we now use the full probability search in both directions.

7.3 Real-Time Recognition on Commercially Available Hardware

It has long been believed that real-time spoken language systems would require immense amounts of computation that would only be available through the use of special-purpose hardware or large-scale parallel computing systems. Unfortunately, in several attempts to design or use parallel or special-purpose hardware at DARPA research sites, none has achieved real-time.

Instead, we believe that it is fundamentally more fruitful to reduce the computation required for spoken language understanding. In particular, rather than gaining a one-time factor of 10 for special-purpose or parallel hardware, we can often produce several orders of magnitude in speed improvement by the invention of clever algorithms that approach the same problem in a different way. (We have described several such algorithms earlier in this report.) In addition, once the computation has been reduced, the algorithms can be run on a broad class of commercially available workstations. This outcome is clearly preferable to requiring special hardware.

7.3.1 Introduction

One goal of the Spoken Language System (SLS) project is to demonstrate a real-time interactive system that integrates speech recognition and natural language processing. We believe that currently, the most practical and efficient way to integrate speech recognition with natural language is using the N-Best paradigm, in which we find the most likely whole-sentence hypotheses using speech recognition together with a simple language model, and then filter and reorder these hypotheses with natural language. The accurate computation of the N Best sentences requires significant amounts of computation.

Several sites have been developing hardware to accelerate the speech recognition process. Two alternative techniques have been special purpose high speed VLSI hardware [68] and several custom boards with more general purpose processors operating in parallel [17] [83]. With the rapid changes in commercial hardware in the past few years we felt that if we could achieve real time with commercially available hardware we would have progressed further. Therefore the goal of this effort was to find appropriate hardware and to improve the algorithms such that the available hardware was sufficient for the problem.

Specifically, our goals are:

1. Real-Time processing with a short delay
2. reasonably inexpensive commercially available hardware
3. source code compatibility with our research programs
4. computation of the N Best sentences for large N
5. using a robust fully-connected statistical grammar
6. within practical memory limitations
7. with negligible loss of accuracy due to real time limitations
8. extendable to larger vocabulary

The new algorithms used in this effort have been described above. Specifically, a method for efficient decoding with a statistical grammar, the Word-Dependent N-Best search and the Forward-Backward Search. In this section, we will describe the class of hardware that we have used to achieve real time recognition.

7.3.2 Hardware

The advantages of using commercially available hardware—if possible—are obvious.

1. No research development cost
2. Improvements in the computer hardware industry are available immediately.
3. Developments can be shared with a broader community.

We will review the sequence of considerations and decisions for the type of hardware that we would use that took place during the course of this project. Briefly, we started out using five separate processor systems for the sampling, signal processing and vector quantization, recognition search, and understanding. As time went on, it became advantageous to eliminate these boards, one by one, until we arrived at the point where we now perform all of the computations directly on a single workstation. This evolution took place because

1. We continued to decrease the computation needed for each phase of the recognition process.
2. Workstations continued to get faster and cheaper, and eventually included built-in audio boards with A/D capability.

The detailed evolution is outlined below.

Starting in January of 1990, we considered several options for off-the-shelf processing boards. It was already quite straightforward to perform signal processing in real-time on boards with signal processor chips. However, the speech recognition search requires a large amount of computation together with several MB of fast readily accessible memory. In the past there have not been commercially available boards or inexpensive computers that meet these needs. However this has been changing over the past couple of years. The Motorola 88000 and Intel 860 chips are now in boards with substantial amounts of random access memory. Most chips now come with C compilers, which means that the bulk of development programs can be transferred directly. If needed, computationally intensive inner loops can be hand coded.

After considering several choices we chose boards based on the Intel 860 processor. The Intel 860 processor has a peak speed of 80 MFLOPS. At the time we considered this processor, most C programs that we had run on both of these machines ran about five times faster than on a SUN 4/280.

Figure 7.7 illustrates the hardware configuration that we envisioned. The host was a SUN 4/330. The microphone was connected to an external preamp and A/D converter which connects directly to the serial port of the Sky Challenger. The Sky Challenger with dual TMS320C30s was used for signal processing and vector quantization (VQ). The SkyBolt was to be used for the speech recognition N-Best search. The boards would communicate with the host and each other through the VME bus, making high speed data transfers easy. However currently the data transfer rate between the boards is very low. The SUN 4 would control the overall system and also contain the natural language understanding system and the application back end.

The design given above met our requirements for speed and flexibility. We would use all four processors during most of the computation. When speech has started the MTU A/D would filter and digitize the signal and feed it—one sample at a time—to the

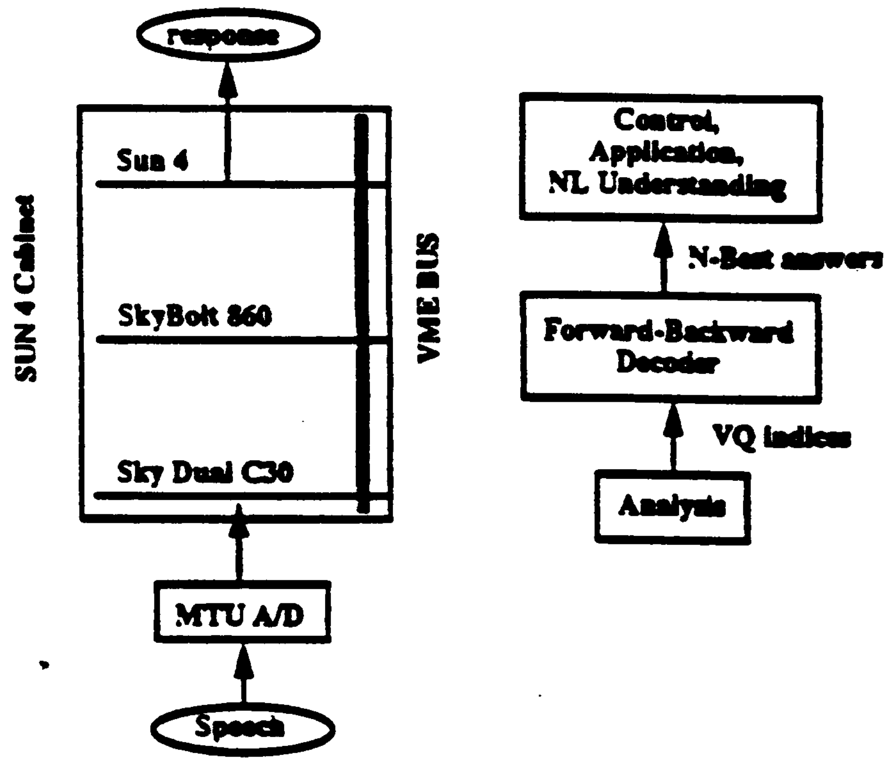


Figure 7.7: Real-Time Hardware Configuration. The Sky Challenger Dual C30 board and the Intel 860 board plug directly into the VME bus of the SUN 4.

C30 board, which would compute the signal processing and VQ in real-time. The SUN 4 would accumulate the speech for possible long-term storage or playback. Meanwhile, the Intel 860 would compute the forward pass of the forward-backward search. When the end of the utterance was detected, the SUN would give the 1-Best answer to the natural language understanding system running on another workstation for parsing and interpretation. Meanwhile the Intel 860 would search backwards for the remainder of the N Best sentence hypotheses. These should be completed in about the same time that the NL system requires to parse the first answer. Then, the NL system can parse down the list of alternative sentences until an acceptable sentence is found.

The computation required for parsing each sentence hypothesis is about 1/2 second. The delay for the N-Best search is about half the duration of the sentence.

7.3.3 Recognition Search on the SUN 4

However, shortly after considering this hardware option, several developments changed our minds as to the most efficient way to proceed.

1. As we sped up the implementation, we noticed that the SUN 4 version got faster, but the Intel 860 version did not. Presumably this was due to better usage of the cache in the SUN 4. The eventual speed advantage over a SUN 4/280 was now only a little over 3.
2. The C compilers for the boards always had bugs, and were never fully integrated into the machine.
3. With each new workstation, the bus architecture changed making it necessary to search for a new accelerator board.
4. New workstations came out that were much faster and much cheaper. The SUN 4/330 was already 1.6 times faster than the SUN 4/280, and when the Sparc II came out it was about 3 times faster than the SUN 4/280. There are other workstations like the SGI 4D/35 and the HP 750 that are faster still. Therefore, the speed advantage of the Intel 860 disappeared completely. In addition, the cost of these new workstations decreased to the point where they were considerably less expensive than the separate processor boards.

Therefore, we quickly abandoned the idea of using a separate board for the recognition search, in favor of simply using the workstation itself. We continued to use the Sky Challenger for front end signal processing for another year, since this replaced a significant amount of computation that could be done easily on a separate board.

7.3.4 Software-Only System

Even the boards used for signal processing have two significant disadvantages:

1. They often cost as much as the workstation they are plugged into.
2. The interface between each board and workstation is complicated, and always different for each combination of workstation and board.

In a separate, internally-funded BBN effort, we have embarked on a productization of our BYBLOS speech recognition technology. One objective of this effort is to make speech recognition available to a broad base of users at an affordable cost. To that end, we have eliminated the disadvantages given above by developing signal processing algorithms that are able to operate in real-time on COTS workstations without requiring additional add-on hardware and without decreasing recognition speed and accuracy. An additional advantage is that we are able to benefit from the improvements in workstation price and performance, with very minimal porting effort. The result of this productization effort has been the development of a real-time system, the BBN RUBY^{TM} speech recognition system.

The RUBY system is based entirely on a single workstation. It used the SGI 4D/35, which has a built-in high quality A/D. The computation needed for the signal processing was reduced to the point where it required only a small part of the available computational power of the workstation. This eliminated the need for a separate hardware board for signal processing.

Our RUBY system was demonstrated at the DARPA Spoken Language Workshop at Arden House in February of 1992. Two applications were demonstrated: recognition of aircraft identification spoken by an air traffic controller and the Airline Travel Information System (ATIS), which included both speech recognition and natural language understanding on the same machine.

7.3.5 Speed and Accuracy

When we started our search optimization effort in January 1990, our unoptimized time-synchronous forward search algorithm took about 30 times real time to recognize with the word-pair grammar with the beamwidth set to avoid pruning errors. The class grammar required 10 times more computation. The exact N-Best algorithm required about 3,000 times real time to find the best 20 answers. When we required the best 100 answers, the program required about 10,000 times real time. Since then we have implemented several search algorithms, optimized the code, and used newer, faster workstations. The forward pass decoder now runs in real time and the N-Best pass now runs in about 1/4 real time, with essentially no loss in accuracy. Below we give each of these methods along with the factor of speed gained.

Statistical grammar algorithm	5
Word-Dependent N-Best	5
Forward-Backward Search	40
Code Optimization	8
Newer Workstations	5
Total reduction in computation	40,000

As can be seen, the three algorithmic changes accounted for 1,000 of this amount, while the code optimization and faster processor accounted for a factor of 40.

Accuracy

It is relatively easy to achieve real time if we relax our goals for accuracy. For example, we could simply reduce the pruning beamwidth in the beam search and we know that the program speeds up tremendously. However, if we reduce the beamwidth too much, we begin to incur search errors. That is, the answer that we find is not, in fact, the highest scoring answer. There are also several algorithms that we could use that require less computation but increase the error rate. While some tradeoffs are reasonable, it is important that any discussion of real-time computation be accompanied by a statement of the accuracy relative to the best possible conditions.

In our most recent real-time demonstration in the ATIS domain, we measured the increase in error rate under real-time operation. We found that the overall error rate increased by only 7% for the answerable queries. Thus, we have shown that the loss due to real time processing is almost insignificant. We believe that this marks the first time that real-time recognition for 1,000 words in continuous speech could be accomplished with so little loss relative to the very best performance possible.

In general, we are confident that the general approach of modifying algorithms and using general purpose workstations for real-time operation will scale up to larger vocabularies and more complex grammars.

7.3.6 Conclusion

We have achieved real-time recognition of the N-Best sentences on a commercially available workstation. Most of the increase in speed came from algorithm modifications rather than from fast hardware or low-level coding enhancements, although the latter improvements were substantial and necessary. All the code is written in C so there is no machine dependence. All told we sped up the N-Best computations by a factor of 40,000 with a combination of algorithms, code optimization, and faster hardware.

7.4 Demonstrations

During the course of this contract, we have developed several demonstrations of speech recognition and understanding. Each one was a breakthrough at the time. Briefly, the major demonstrations were:

1. August 1990: Near-real-time speech recognition of Resource Management domain. This demonstration was faster than the only other demonstration at the time, which used a special-purpose three processor board for the search.
2. January 1991: Real-time spoken language system. We connected the real-time speech recognition to the natural language system running a database query system in the DART TRANSCOM Military Transportation Logistical Planning Domain.
3. February 1992: Real-Time spoken language on a single workstation. We demonstrated a full spoken language system in the ATIS domain. The demonstration used no extra hardware beside a single SGI workstation. Even so, it was two to four times faster than any of the other demonstrations in this domain, even though they all used additional hardware for the front end signal processing. In addition, the accuracy of this demonstration significantly exceeded the others, partially because of the capability of using a trigram grammar in real time.
4. February 1992: Demonstration of a real air traffic controller speech recognition application. The application was significant in its extremely high accuracy, its capability for rejecting out of vocabulary items, and its ability to recognize and output a subgrammar embedded in a sentence within 300 ms of the end of speaking the subgrammar phrase.

These demonstrations are described in more detail below.

7.4.1 August 1990: Real-Time Speech Recognition

In August of 1990, we demonstrated the system described in June of 1990 at the Hidden Valley Workshop. In particular, we used a SUN 4/330 workstation, with a Sky Challenger signal processing board for the front end signal processing, and an MTU A/D board for capturing the signal. All of the search computation was performed on the SUN workstation. We demonstrated near real-time recognition in the Resource Management Domain. By "near real-time", we mean that the first choice answer for the sentence was typed within a very short delay of when the speaker stopped speaking. Then, the N-Best sentence hypotheses were typed within a second or two after that. The demonstration of speaker-independent recognition had been trained on only a small number of speakers. In particular, 8 male and 7 female speakers. (We had previously demonstrated at the June '90 workshop that we had obtained approximately the same accuracy by training with a large amount of speech from a small number of speakers as with a small amount of speech from a large number of speakers.) We believe that this was the first real-time demonstration on the RM domain.

In all, we sped up the search by a factor of 20,000 with a combination of hardware and software improvements. A real-time recognition system also required implementing a complete front end that would filter, sample, analyze, and vector quantize the speech, and pass the results on to the recognition search—all in real time. We used a programmable MTU filter and A/D converter to do the basic speech sampling. We used a Sky Challenger with two TMS320C30 processors to control the MTU and to perform the signal processing (Mel-Frequency Cepstral Analysis) and vector quantization. The Sky Challenger was placed on the VME bus of a SUN 4/330 workstation. Our plan was to use the SkyBolt signal processing board for the recognition search. This would also be connected directly to the same VME bus, and the vector quantization output from the C30 processors would be fed directly into the Intel 860s on the SkyBolt. However, there were problems with the C compiler on the SkyBolt that had a large enough memory to run the recognition program. Therefore, for the time being we ran the recognition directly on the SUN 4 workstation. We found that, since we had sped up the recognition so much, the recognition was able to run in almost real time on the SUN 4 workstation without the need for further accelerator boards!

We implemented a demonstration of speech recognition in which the speaker says a sentence when prompted to do so. While the speaker is speaking, the signal processing and vector quantization is being performed in real time. The forward pass recognition search is also proceeding, as fast as possible. Shortly after the speaker stops speaking, the system has finished the recognition of the most likely sentence, and prints the answer onto the screen. It plays the speech back to the user so that s/he can verify that the answer is correct. (In a spoken language system, this answer will be fed to the natural language component for understanding.) Meanwhile, the system performs a backward search for the N-Best sentences using the forward pass to speed up the search tremendously. The top N sentences are then displayed on the screen—with their corresponding acoustic and statistical language model scores. The backward pass is fast enough so that it is always finished long before the sentence has been replayed to the speaker. (In the spoken language system, these N answers will be made available to the natural language component, in case the first choice sentence did not parse or did not make sense.)

We created a speaker-independent speech model using the speech of eight male speakers. The training data of each speaker was first quantized using a male-dependent codebook, and then a speaker-dependent model was estimated for each of the speakers. Finally, the eight models were averaged to create a male-dependent speech model. We repeated the same steps for the 7 females available. The demonstration used a statistical first-order class grammar. We defined 548 word classes. Most words were in their own class, but words that are completely interchangeable, like names of ships and ports, and the digits and days of the week, are grouped together into classes. The perplexity of the resulting grammar was 22. However, all sequences of words are possible, since we assume that any class can be followed by any other with some small probability.

January 1991: Real-Time Spoken Language System

For the next demonstration, in January of 1991 we connected the real-time BYBLOS N-Best recognizer described above to a natural language understanding component and demonstrated a complete real-time HARC spoken language system. The speech and NL processes communicate through an ethernet connection, thus enabling our real-time SLS to run on different types of machines, if desired. Since the communication data rate between the speech and NL components is very low, the ethernet connection does not cause any bottleneck in the integration.

The way the system worked was as follows: As the speaker says an utterance, the speech component first performs the forward pass of the Forward-Backward recognition search. As soon as the speaker has stopped speaking, the most likely sentence hypothesis is sent to the NL component, which begins to parse it. Simultaneously, the speech component performs the backward pass to find the N-Best hypotheses (this backward pass is completed in a fraction of real time). The N-Best hypotheses are then sent to the NL component, which goes through each of the hypotheses until it finds one that is linguistically meaningful. The highest-scoring linguistically-meaningful hypothesis is taken as what was said and is used to perform the database query or command. Typically the response to a query or command appears on the screen within only a few seconds of the end of an utterance.

One other interesting feature of the system is that the user can either speak or type a query at any point.

7.4.2 February 1992: Real-Time SLS using BBN/RUBY

At the workshop, we demonstrated two example systems that employ the BBN RUBY speech recognition system. Both demonstrations run on Silicon Graphics workstations (Personal IRIS 4D/35 and Indigo), which contain a built-in programmable A/D-D/A. The signal processing and vector quantization, which runs in a separate process from the recognition search, communicates with the recognition search via network sockets. We have reduced the computation required for this front end processing to the point where it requires little enough of the CPU so that there is enough left over to perform the more expensive search in real time. Since accuracy is our primary concern, we have verified that this signal processing results in the same accuracy as our previous signal processing software.

Real-Time ATIS System

The ATIS demonstration integrated BBN's DELPHI natural language understanding system with the RUBY speech recognition component. RUBY is used as a black-box, controlled entirely through an application programmers interface (API). The natural language component is our current research system, which runs as a separate process. Both processes run on the same processor, although not at the same time. The NL processing is performed strictly after the speech recognition, since competing for the same processor could not make it faster.

The speech recognition component performs three separate steps. First, it uses a forward Viterbi-like computation to find the 1-Best speech answer in real time as the user is speaking. Immediately after the user stops speaking, it displays the 1-best answer. Then, it performs a backwards N-Best pass to find the N-Best hypotheses. Finally, we rescore each of the text hypotheses using a higher-order n-gram statistical class grammar and reorder the hypotheses accordingly. In this application, we use a trigram model; this rescoring computation requires very minimal processing. (Note that at this time we have omitted the acoustic rescoring stage in which we could rescore with between-word triphone models and semi-continuous HMM models.) After that, the N-Best answers are sent to DELPHI which searches the N-Best answers for an interpretable sentence and then finds the answer to the question.

Normally, the system displays the 1-Best answer within a half second of detection of the end of speech. This is fast enough that it feels instantaneous. (This speed is in marked contrast with the other near real-time demonstrations shown at the workshop, which all required from two to five times real time, resulting in a delay that was at least equal to the length of the utterance—usually several seconds.) Next it performs the N-Best recognition, and then interprets the answer. The N-Best recognition usually runs in less than 1 second, since it is sped up immensely by the forward pass. The rescoring of the N-Best hypotheses with trigram grammar requires essentially no time. The time required for the interpretation depends on how many of the speech answers must be considered, and on how complicated a retrieval results. In most cases, this phase requires only another second or two.

To operate the demonstration system, the user clicks on the "Push to Talk" window at the top of the screen. The status will change from "ready" to "listening". As soon as the user begins speaking, the status will change to "beginning of speech". When (s)he stops speaking, it will change to "end of speech". The system briefly displays its status as "First-Best" and "N-Best" while it completes these phases of the recognition. Finally, the system will "Interpret" the query, which includes all parsing, semantic interpretation, discourse modeling, and data retrieval from the actual database.

The answer displayed in the speech window first contains the answer from the 1-Best pass, then the top-choice of the N-Best, and finally the sentence chosen by DELPHI. The N-Best hypotheses are displayed at the bottom of the screen for information only. Then, the answer to the query is displayed under the recognized sentence. If the answer to be displayed is larger than will fit in the window, it can be scrolled. A history of the previous four sentences are shown in a window that can scroll all the way back through the previous questions. If the user wishes to review any of the previous answers in more detail, they may mouse on the arrow to the right of the question, which brings back a copy of the question and answer in a separate window that may be placed and sized as desired, and then used for reference as long as needed.

To the right of the main display, we also display the *discourse state*, which consists of the set of constraints that were used to answer the query. In this way, the user can verify how much of the previous context was actually used to answer the question. As each successive query is interpreted, the system may add new constraints, modify old ones, or completely reset the context. The user may also reset the discourse state by speaking any of the commands, "BEGIN SCENARIO", "END SCENARIO", or "NEW SCENARIO".

Real-Time Speech Recognition for Air-Traffic Control Applications

At the February 1992 DARPA Workshop, we also demonstrated the BBN RUBY system configured for air-traffic control (ATC) applications. The demonstration was developed under a separate, government-funded effort. This system is notable for its high speed, accuracy, robustness, and reliability, all necessary qualities for ATC applications requiring human-machine interaction. Such applications include training systems, where the trainee controller interacts with a simulated world, and operational environments, where a controller's interaction with a pilot could automatically generate a database retrieval request for flight plan information.

In this demonstration, the system extracts the aircraft flight identification from an utterance as soon as that information has been spoken. For example, if the controller says, "Delta three fifty seven descend and maintain 2000", the flight information could be captured for display or other uses by the time the controller has said the first syllable of "descend". To achieve this immediate response, the speech recognition detects when the controller has completed the flight identifier and is speaking the rest of the utterance. This requires a different process than is usually used for speech recognition. Normally we wait until the end of the sentence to determine the most likely word string for the complete utterance. For this application, the system stops the recognition process as soon as it determines that it is most likely to have the complete flight information.

Another unique capability that is demonstrated here is the capability to reject the flight ID if it is not in a specific closed set. Again, this is done by explicitly modeling the likelihood that the user has spoken a flight ID other than the set that is expected at any given time.

Chapter 8

Detecting and Adding New Words

Another major area of work during this contract was related to the use of new words, that is, words outside the initial vocabulary of the system. In a large vocabulary system, the user will invariably use words that are not in the current vocabulary. As described below, the usual system behavior will be quite confusing and frustrating for the user. Therefore, it would be quite useful if the system could alert the user when a new word was spoken—even though it could not recognize the new word. Once the need for a new word has been identified, the user must add this word to the system without stopping to collect additional speech training data or restarting the whole system. The issue is to define all the necessary information for the new word. These include its orthographic spelling, its phonetic spelling, and where it fits in the language model.

In this effort we developed what we believe to be the very first method for detecting new words. We also developed a novel method for adding a new word to the system's vocabulary. Details are given below.

8.1 Detecting New Words

8.1.1 The New-Word Problem

Current continuous speech recognition systems are designed to recognize words within the vocabulary of the system. When a *new word* is spoken, the systems recognize other words that are in the vocabulary in place of the new word. When this happens, the user does not know that the problem is that one of the words spoken is not in the vocabulary. He assumes that the system simply misrecognized the word, and therefore he says the sentence again and again. Current systems do not tell the user what the problem is, which could be very frustrating.

Adding the ability to detect new words automatically is desirable and improves the performance of the system. Once a new word is detected, it is possible to add the word to the vocabulary with some extra information from the user, such as repeating the word within a carrier phrase and typing in the spelling of the word.

8.1.2 Approach

An obvious zero-order solution for detection of new words problem is to apply some rejection threshold on the word score. If the score reaches a level higher than the threshold then a new word is detected. However, when we examined the scores of words in a sentence, we found that the score of correct words varies widely, making it impossible to tell whether a word is correct or not. Therefore, this approach for detecting new words did not work well.

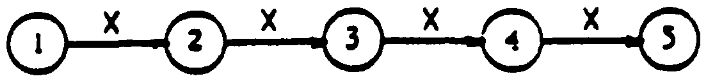
Our proposed solution is to develop an explicit model of new words that will be detected whenever a new word occurs. The word model should be general enough to represent any new word. It should score better than other words in the vocabulary where there is a new word; it must always score worse on words in the vocabulary than the model of that word. Given the above assumptions, we tried four acoustic models of new words which are described below.

Proposed Models For A New Word

All new-word models consist of sequences of phonemes. Each phoneme is represented by a 3-state Hidden Markov Model (HMM). The states are connected from left to right with "self-loops" on each state. There are transition probabilities on these connections. Also, associated with each state is a spectral distribution for the VQ clusters (256 clusters).

The first new-word model we tried consisted of four phonemes (see figure 8.1). It has 5 states and 4 identical phonemes with an adjustable flat spectral distribution. From now on we'll refer to this model as M1.

The second new-word model that we tried was a model that allows for any sequence of phonemes of at least two phonemes long (see figure 8.2a). The model has 3 states,



X: phoneme HMM with 3 flat spectral distributions

Figure 8.1: Flat New-Word Model (M1).

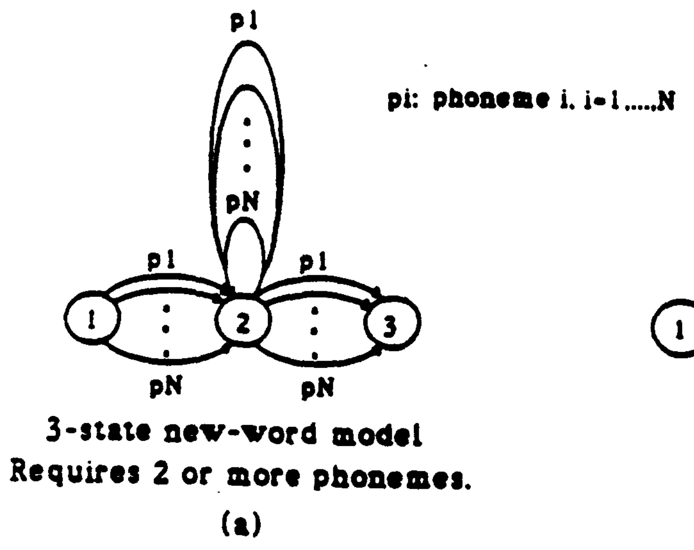
all phonemes in parallel from the first state to the second state, all phonemes in parallel from the second state to the third state and all phonemes in parallel looping on the second state. The model has 3N phoneme arcs, where N is the number of phonemes used in the system (N=53 in our system). All phonemes are represented with context-independent phoneme models. Note that this is in contrast to the normal vocabulary of the system, which uses context-dependent phoneme models. The context-independent phoneme models in the new-word model are trained on the same data as the system vocabulary. We'll refer to this model as M2.

The third new-word model we used was similar to the second model (see figure 8.2b). It has 5 states with a minimum of 4 phonemes. The model has 5N phoneme arcs. All phoneme models are context independent. We'll refer to this model as M3.

The fourth word model was a diphone word model (see figure 8.3). It consisted of models of the phonemes in the context of the previous phoneme (diphones). It allows for any sequence of diphones with a minimum of two diphones. This model has N+2 states, N is the number of phonemes used in the system and the 2 for the beginning and end word boundaries. It has N^2 + 2N left-context phoneme arcs. We'll refer to this model as M4.

Grammar

We used a first order statistical class grammar. A statistical class grammar consists of class nodes, word arcs and arcs between classes. Each class node has a number of word



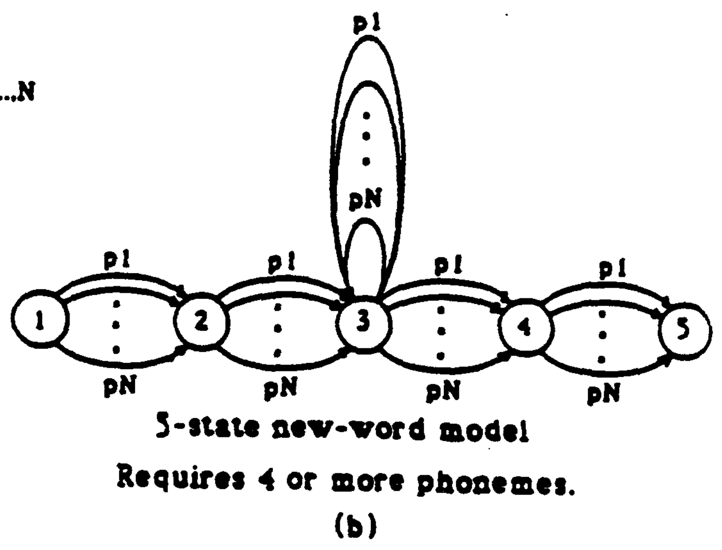
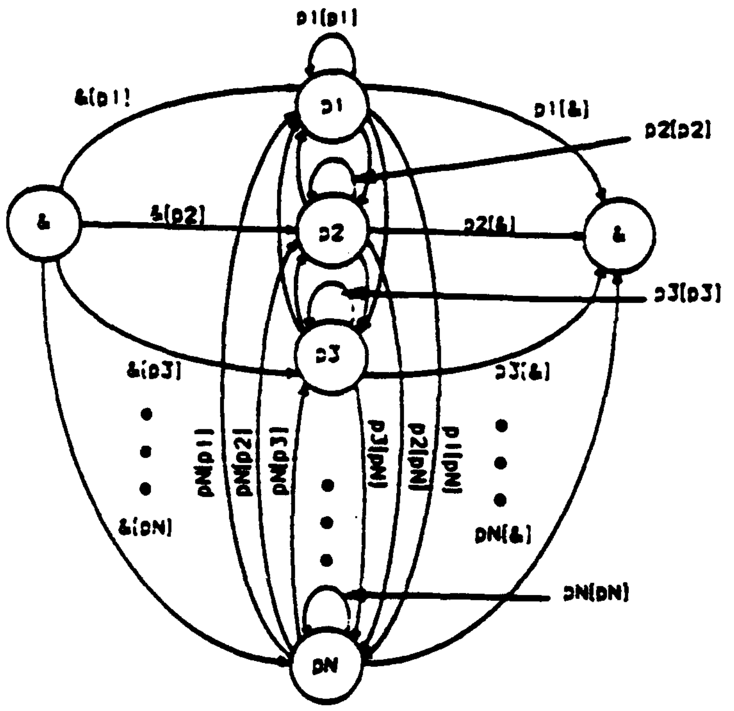


Figure 8.2: 3-State and 5-State New-Word Models (M2 and M3).

arcs emerging from it. Word arcs in the same class have equal probabilities. There are transition probabilities between the classes that depend on the training given to the class grammar.

New words are more likely to appear in open classes than closed classes. Open classes are the classes that accept new words (e.g., ship names, port names) as opposed to closed classes that do not accept new words (e.g., months, week-days, digits). We created a separate new-word model for each open class. Also, it is easy to add the open class words to statistical class grammars and to Natural Language syntax and semantics.



&	word boundary
pi	phoneme i, i=1.....N
pi[pj]	phoneme j preceded by phoneme i.

Figure 8.3: Diphone New-Word Model (M4).

8.1.3 Experiments and Results

The experiments we describe here use the DARPA 1000-word Resource Management Database for continuous speech recognition and BYBLOS, the BBN continuous speech recognition system[31]. The database is limited to 1000 words and does not include data to test for new words. Therefore, we simulated new words in the system simply by removing words from the 1000 word vocabulary that occur in the test sentences.

In the following we give results for experiments that used the four word models for a new word. The experiments were run on 7 speakers from the speaker dependent portion of the database (BEF, CMR, DMS, DTB, DTD, JWS and PGH), 25 test sentences per speaker. We created a statistical class grammar from the remaining words in the vocabulary. We varied the perplexity of the statistical class grammars simply by changing the number of training sentences. A bias *for* new words was implemented in the case of the flat new-word model (M1); a bias *against* new words was implemented to reduce the false alarm rate in the 3-state, 5-state and diphone models (M2, M3 and M4). The bias is a scaling factor which is multiplied by the new word arc probability from an open class, reducing the probability of selecting the new word from that class with respect to the rest of the words in the class.

We ran experiments to detect new words from the classes *ship name* (e.g., Downes) and *ship name possessive* (e.g., Downes's). In these experiments we removed from the vocabulary 45 ship names and their corresponding possessives (90 words). The 175 test sentences included 59 occurrences of new ship names. Also, we ran an experiment on detecting new words from the class *port name*. In this experiment we removed 11 port names from the original vocabulary and the test sentences had one occurrence of each. Then, we ran experiments on detecting new words from 7 different classes at the same time. In these experiments we removed a total of 41 words from the original vocabulary from the classes *ship name* (10 words), *ship name possessive* (10 words), *port name* (8 words), *water name* (5 words), *capability* (4 words), *land name* (4 words). We included the class *track name* but did not remove words from this class because the test sentences did not have any words from this class. The 175 test sentences have 62 occurrences of new words.

Using the model M1 we ran an experiment to detect new words from the classes *ship name* and *ship name possessive* without distinction between the two classes. The results showed no detections or false alarms. When we tuned the results with a bias *for* the new-word model, the detection rate was 67% and the false alarm rate was 51%. This model is not useful since it has a very high false alarm rate.

We ran experiments using the model M2 and the results are shown in Table8.1. A bias *against* the new-word models was used to reduce the false alarm rate. The columns in the table are described below, then followed by an example as an illustration.

- classes: the classes of new words allowed.
- perp: the perplexity of the grammar.
- cor: correct or exact detection rate as a percentage of number of new words. The removed word was exactly replaced by the *new word* model of the same class. In the following example, the lines marked REF is what was actually spoken; the lines marked HYP is what was recognized by the system. The sentence appears on more than one line for clarity. The words LAMPS, of class *capability*, and MOZAMBIQUE, of class *water name*, were removed from the vocabulary. The system recognized NEW-CAPABILITY and NEW-WATER-NAME, respectively, exactly in their places.

```
SENTENCE (1237)
REF: how many LAMPS
HYP: how many NEW-CAPABILITY

REF: cruisers are in
HYP: cruisers are in

REF: MOZAMBIQUE     channel
HYP: NEW-WATER-NAME channel
```

- cls: close call or close detection rate. That is, the *new word* was detected but there was an insertion or deletion in its vicinity. The new-word HAWKBILL, of class *ship name*, was recognized correctly as NEW-SHIP-NAME, but the word due DUE was deleted.

```
SENTENCE (0464)
REF: when+s HAWKBILL      DUE in
HYP: when+s NEW-SHIP-NAME *** in

REF: port
HYP: port
```

- sw: switch between classes, i.e., the new word was detected, but was assigned to the wrong class. In the following example the word PEORIA is a *ship name* and the system recognized it as a *ship name possessive*.

```
SENTENCE (1006)
REF: when was PEORIA          last
HYP: when was NEW-SHIP-NAME+S last

REF: in the atlantic ocean
HYP: in the atlantic ocean
```

- det: total detection rate, sum of cor, cls and sw. This is the rate of correctly detecting the existence of a new word in the vicinity of its occurrence. (While we would like the system to detect the exact location and the class of the new words, it is also useful to simply detect that a new word has occurred).

- fal: false alarm rate, percentage of number of false alarms to the total number of test sentences. A false alarm is a *new word* detected where there was no new words in that part of the test sentence. In the following example the word AREA was misrecognized as NEW-TRACK-NAME, which is a false alarm for a word from the *track name* class.

```
SENTENCE (1025)
REF: WHEN   DID sherman last
HYP: WHEN+S THE sherman last

REF: downgrade for asuw
HYP: downgrade for asuw

REF: mission AREA
HYP: mission NEW-TRACK-NAME
```

classes	perp	cor	cls	sw	det	fal
shipname(+s)	100	42	36	5	83	1.7
shipname(+s)	60	49	30	5	84	1.1
portname	100	27	37	-	64	0.6
7 classes	100	44	6	24	74	3.4

Table 8.1: Detection of new-words results using the model M2.

For the word model M2, the first experiment was detecting new words from the classes *ship name* and *ship name possessive*. The perplexity of the grammar was 100. We had a detection rate of 83% and a false alarm rate of 1.7%. In the second experiment we changed the perplexity of the grammar to 60 to measure the effect of the perplexity on the detection rate and the false alarm rate. There was no significant difference in the detection rate (84%) but the false alarm rate dropped to 1.1%, a reduction of 35% in the false alarm rate. Our third experiment was detecting new words from the class *port name* with grammar of perplexity 100. We had a detection rate of 64% and a false alarm rate of 0.6%. In the fourth experiment we tried to detect new words from 7 different classes, with a grammar of perplexity 100, the detection rate was 74% and the false alarm rate was 3.4%.

In Table8.2, we compare the detection results using the new-word models M2, M3 and M4. The experiments were run on detecting the same set of removed words from 7 classes with a grammar of perplexity 100, and similar bias against the new words.

model	cor	cls	sw	det	fal
M2	44	6	24	74	3.4
M3	37	8	26	71	4.0
M4	21	14	41	76	8.6

Table 8.2: Detection of new words results for M2, M3 and M4 with 7 new-word classes and grammar of perplexity 100.

The results for the model M2 are the same as the last row in Table8.1. The results for the model M3 are 71% detection rate and 4% false alarm rate. As for the model M4, the results are 76% detection rate and 8.6% false alarm rate.

From Table8.2 we can say that M2 is the best new-word model because it has the lowest false alarm rate and a high detection rate. M3 results are very close to M2 results, but M2 outperforms M3 in all categories. M4 has the highest detection rate but it has also the highest false alarm rate. The high false alarm rate is due to the fact that the diphone model M4 matches the existing words much better than the 3-state model M2. That is, even though the diphone model is somewhat better for new words, it is a much better model for existing words because it has been trained on the existing words. In fact, it matches existing words better than their own models because it does not have the sequential constraints that are in the models of the existing words. Therefore, it is hard to tune the bias such that the diphone model M4 detects only new words. In addition, the diphone model requires much more computation, since it consists of all possible diphones.

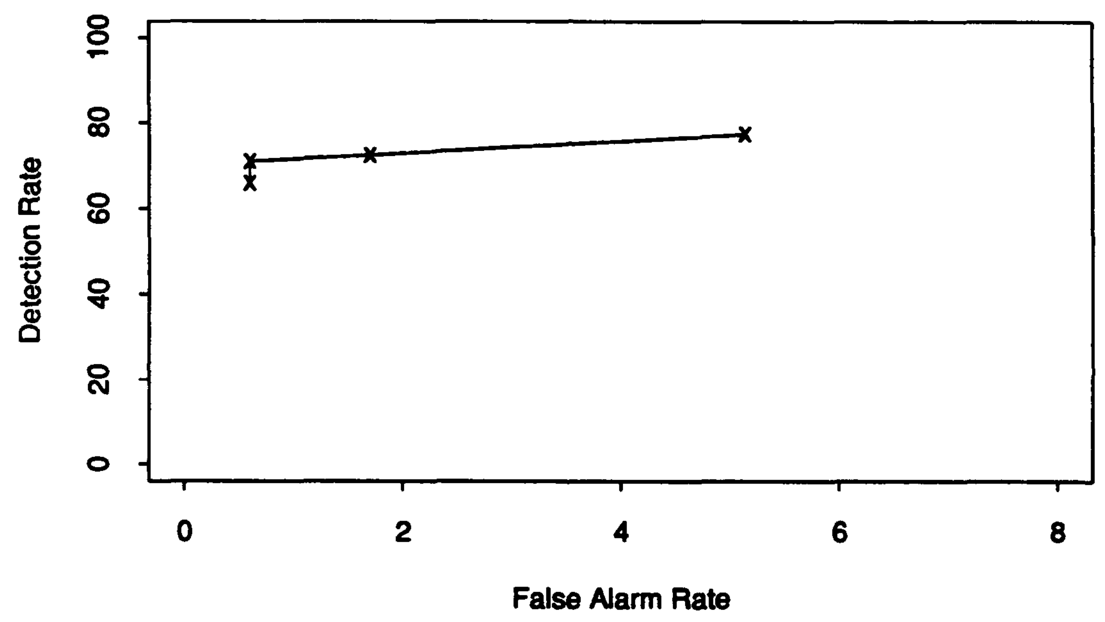


Figure 8.4: Detection Rate vs. False Alarm Rate for the Speaker-Dependent Part.

8.1.4 Speaker-dependent vs Speaker-Independent Models

To further test our new-word detection algorithm, we ran experiments on the speaker-dependent and speaker-independent portions of the DARPA 1000-Word Resource Management Corpus. In these experiments we tested the detection of new words from the classes, *ship-name, ship-name-possessive, port-name, water-name, land-name, capability and track-name*. The results were obtained by training the system on 500 sentences which do not include any token from the new words that are in the test sentences. A statistical class grammar of perplexity 100 was used in obtaining these results. The detection rate and the false alarm rate can be tuned by varying the bias against the new words in the statistical class grammar. The bias is a scalar multiplied by the probability of a new word in the class grammar. If we increase the bias against the new words, the false alarm rate decreases with a minimal loss in the detection rate.

In Figure 8.4, we plot the detection rate versus the false alarm rate for the detection of new words at various bias values against new words. The experiments were run on the speaker-dependent part of the Corpus. A good operating point is at the knee of the curve where 71% of the new words in the test sentences were detected and less than 1 sentence out of 100 sentences had a false alarm. This is a very useful result which can be used in various application to detect new words in speech recognition.

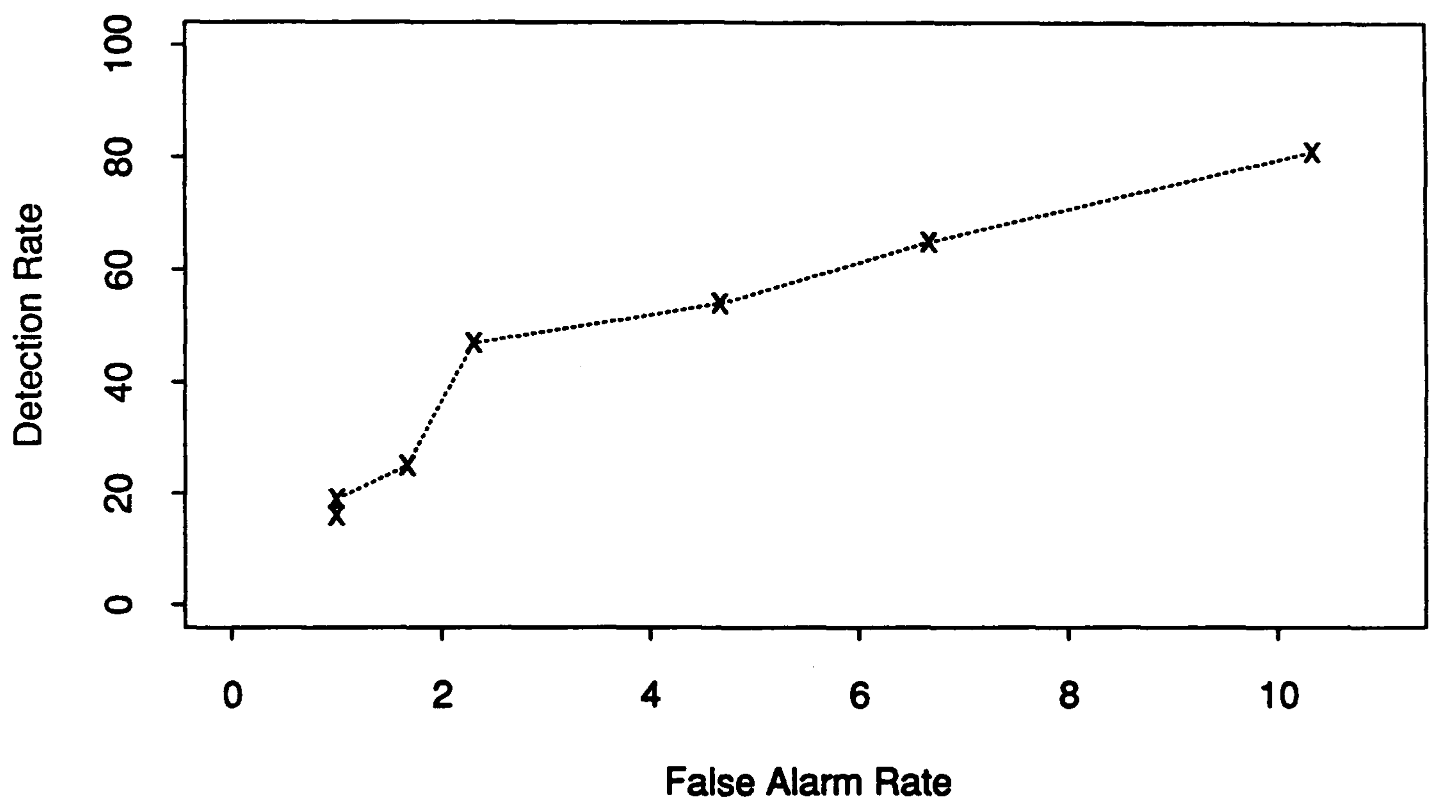


Figure 8.5: Detection Rate vs. False Alarm Rate for the Speaker-Independent Part.

In Figure 8.5, we plot the detection rate versus the false alarm rate for the detection of new words at various bias values for the speaker-independent part of the above corpus. The system was trained using the 12-speaker speaker-independent training paradigm (12-SI) [57] and a statistical class grammar of perplexity 23. The best operating point is a detection rate of 50% and a false alarm rate of 2%.

Comparing Figures 8.4 and 8.5, we clearly see that this solution for the detection of new words works better for the speaker-dependent scenario than the speaker-independent scenario, especially when we consider that the perplexity of the grammar used in the speaker-independent experiment was lower.

Conclusions

From the above results we have shown that the problem of detecting new words can be solved by selecting an explicit word model for new words. We tried 4 models for new words and compared their results. The 3-state model, consisting of two or more context-independent phonemes, has a high detection rate of 74% and the lowest false alarm rate of 3.4%. The 5-state model did not show any advantages for increasing the minimum length of new words to 4 phonemes. The 3-state model outperformed it in all aspects. The diphone model had a high false alarm rate because it models the existing words very well. Reducing the perplexity of the class grammar from 100 to 60 does not affect the detection rate significantly but reduces the false alarm rate.

A second and expected conclusion is that the detection of new words is easier for speaker-dependent models than for speaker-independent models. Interestingly, the false alarm rate seems to be closely related to the overall word error rate.

8.2 Adding New Words

The new-word problem in speech recognition systems has two parts. The first part is to detect that the user has spoken a new word. The second part is to add that new word to the vocabulary of the system after it is detected. The two parts can be solved independently. Some speech recognition systems have a capability of adding new words. However, to the best of our knowledge, the BBN BYBLOS continuous speech recognition system is the only system that has a capability of detecting new words.

In this section we present an interactive technique for modeling and adding new words to the vocabulary of the system.

8.2.1 Modeling a New Word

Once a new word is detected, it is desirable to add the word to the vocabulary of the system. The system needs a phonetic transcription for the new word in order to build a Hidden Markov Model (HMM) for the new word.

It is assumed that the user does not know more that how to spell or pronounce the new word. When the system asks the user to type the orthographic spelling of the new word, the system will check if the word is in fact a new word. Then, it will look it up from a large phonetic dictionary. If the system finds the phonetic transcription in the dictionary, it will use it in building a model for the new word. Otherwise, the system has to obtain a phonetic transcription for the new word using an alternate method.

In the literature, Lucassen *et al.* [60] and Bahl *et al.* [10] have used an information theoretic approach to find the phonetic transcription of a word given its orthographic spelling. Although the method had high accuracy, it needs a very large training dictionary and is very compute intensive.

Below we present several possibilities for finding the phonetic transcription of a new word. First, we present the phonetic recognition capability of the BYBLOS system. Then, we present the phonetic transcription capability of DECtalk. Finally, we present our approach that combines the above two methods to generate phonetic transcriptions that are sufficient for recognition purposes.

Phonetic Recognition

The first approach was to run the speech recognition system in a phonetic recognition mode. The goal was to see if the phonetic recognition accuracy is high enough to be used to transcribe new words.

Unit	Grammar	correct phonemes	error rate
Phonemes	Null	59.8	44.0
Phonemes	CG	69.4	34.9
Diphones	CG	78.8	24.2
Triphones	CG	84.4	18.0

Table 8.3: Phonetic Recognition Performance of the BYBLOS System.

Table8.3 shows the phonetic recognition accuracy of the speech recognition system. The table shows the accuracy for context-independent and context-dependent phoneme models. The context-independent models were tested with a full-branching grammar (null), where each phoneme can be followed by any other phoneme with equal probability, and a statistical class grammar (CG), computed from the phonetic transcriptions of 600 training sentences. Each phoneme is placed in a separate class. As expected, Table8.3 shows that using a class grammar improves the phonetic recognition accuracy. Also, the table shows that using context-dependent models (diphones and triphones) improves the accuracy of the system.

The phonetic recognition accuracy shown in Table8.3 is not high enough to be used for transcribing new words as will be seen below. Hence another method should be used.

DECtalk

The next approach was to use the phonetic transcription capability of DECtalk. We ran several experiments which tested the suitability of DECtalk transcriptions for speech recognition purposes. Table8.4 shows DECtalk's phonetic transcription accuracy for 1000 words from the Resource Management Corpus. Then, DECtalk transcriptions were used in word recognition experiments.

Phonetic transcription source	Correct phonemes	Error rate
DECtalk	88.4	12.5

Table 8.4: Phonetic Recognition Results for RM 1000 Words Using DECtalk's text-to-sound rules.

Phonetic Transcription Source	Word Error Rate
All words hand transcribed	4.4
All words from DECtalk	21.2
41 words from DECtalk	6.2

Table 8.5: Word Error Rates in Speech Recognition Using Several Phonetic Transcription Methods.

Table8.5 shows the error rates on the 1000-Word Resource Management Corpus. The tests were run on 7 speakers from the May 88 test. The table shows the performance of the system when all the words were hand transcribed, all the words transcribed by DECtalk, and finally 41 words, designated as new words, transcribed by DECtalk and the rest hand transcribed.

There was 62 instances of the 41 words in the test sentences. The system recognized them all when the words were hand transcribed. When just these words were transcribed by DECtalk, the system recognized 48 of the instances and missed 14 (22.6%). The errors were in 9 out of the 41 words. 5 words were misrecognized all the time. This clearly shows that relying on DECtalk transcriptions is not sufficient for recognizing the new words.

Probabilistic Transformation of DECtalk transcriptions

To get a more reliable phonetic transcription for a new word we probabilistically combine two knowledge sources to obtain an improved phonetic transcription. The system prompts the user to type the orthographic spelling of the new word. The system passes the spelling to DECtalk which in turn produces an initial (possibly errorful) phonetic transcription for the new word. Then we use a probabilistic transformation method to create a pronunciation network. Then the system prompts the user to pronounce the word and runs in a phonetic recognition mode. The pronunciation network is used to constrain the phonetic recognition process. Figure 8.6 summarizes this phonetic transcription method.

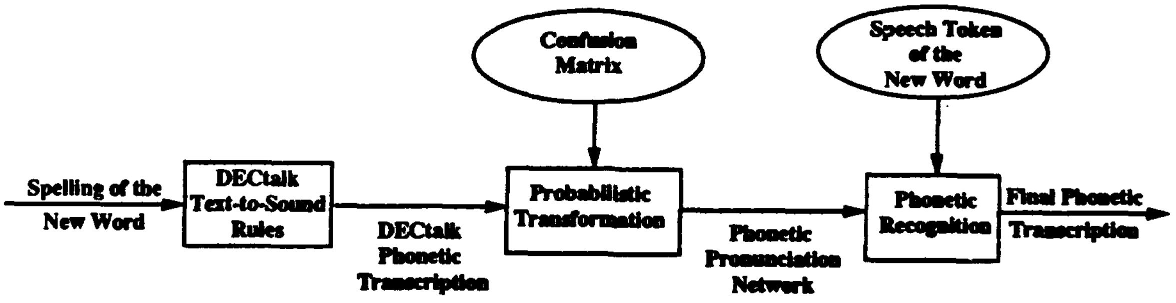

Confusion Matrix
Speech Token of the New Word
Spelling of the New Word
DECtalk Text-to-Sound Rules
DECtalk Phonetic Transcription
Probabilistic Transformation
Phonetic Pronunciation Network
Phonetic Recognition
Final Phonetic Transcription

Figure 8.6: Obtaining a Phonetic Transcription for the New Word Given its Orthographic Spelling.

The probabilistic transformation is performed as follows. Starting with a set of correct transcriptions of a large number of words, we find the corresponding phonetic spellings given by DECtalk and we compute a confusion matrix between the two sets of phonemes. The confusion matrix is then normalized, resulting in the probability that DECtalk will confuse each phoneme for each of the other phonemes. Given DECtalk's phonetic transcription for a new word, we then form a finite-state network that contains all the possible pronunciations of the new word, given the confusion probabilities. Using this network to constrain the possible phonetic sequences for the new word, we perform phonetic recognition of the single token of the pronunciation of the new word provided by the user. The phonetic sequence that results in the highest score is then our final, corrected phonetic transcription of the new word.

Note that using BYBLOS to perform phonetic recognition on the single token does not result in sufficiently high accuracy for our purposes. The procedure described above uses a very tight pronunciation network, employing another source of information (DECtalk), to constrain the search and, hence, improve the phonetic recognition accuracy. The phonetic transcription will be used in building a specific model for the new word.

The total instances of new words that were not recognized by the system after their transcriptions were improved dropped from 14 (22.6%) to 2 (3%). Table8.6 shows the word error rate when we used improved transcriptions for the new words by the above corrective procedure. When compared with the results in Table8.5, it clearly shows that the approach is viable and can be used to transcribe new words.

Phonetic Transcription Source	Error Rate
41 words after corrective procedure	4.5

Table 8.6: Word Error Rates in Recognition After the Corrective Procedure.

8.2.2 Adding a New Word to the System

When a new word is detected and a proper word model has been built for it, the word is added to the dictionary of the system and the word model is added to the word-model database. The new word will be part of the system the next time it is used.

Adding a new word does not end at building a model for the new word. The system has to know how this new word fits in the system language model. The system adds the new word to its statistical class grammar by finding from the user how the new word fits in the language model. To do so, it prompts the user with a list of classes that allow new words and a sample word from each class. The user picks a word (or class) or more that the new word should belong to.

It is assumed that all words within a class, in the statistical class grammar, are equally probable. When adding a new word to one of the classes, the probabilities of the words in that class are reset to $1/(N+1)$, where N is the number of words in the class. Other techniques for adding new words to a language model are suggested by [48] and [56].

8.2.3 Conclusions

In this work we have reported on the detection of new words for the speaker-dependent and speaker-independent paradigms. A useful operating point in a speaker-dependent paradigm was defined at 71% detection rate and 1% false alarm rate. For speaker-independent models, we detected about 50% of the new words with a 2% false alarm rate. We have shown that it is possible for a naive user to add new words simply by typing the word and saying it once. The system can determine a reasonable phonetic pronunciation for the new word by using a combination of the predicted spellings and the spoken example of the word.

Chapter 9

Direct Cooperation with Other Research Sites

One goal of the project is to enable efficient cooperative research to take place at multiple sites. Besides the obvious benefits from sharing our ideas, one of the most profitable modes of cooperation has been in enabling other research sites to use our HMM recognition system as a first step in their research. Specifically, we have cooperated with the research effort at Boston University in using Stochastic Segment Models, the effort at BBN in using Neural Networks, and the effort at Unisys in building the understanding module of a Spoken Language System (SLS).

9.1 Cooperation with Boston University

In an effort to combine the strengths of different speech recognition systems to improve performance, we developed a simple method for combining systems, based on the N-best paradigm developed at BBN. In particular, we are interested in combining the HMM-based system at BBN with another at Boston University which uses stochastic segment models (SSM). The SSM considers each phonetic segment as a single entity, and thus must explicitly consider many different phonetic segmentations to determine the most likely answer. As a result, it requires significantly more computation than the HMM. However, preliminary experimental results indicate that the SSM might be more powerful than the HMM. Even if it turns out not to be more powerful, the information represented in the SSM is different to some extent from the HMM, and therefore could complement the HMM nicely.

To this end, we use a variation of the N-best paradigm, as follows. Given a speech utterance, the HMM system is used to create the N most likely alternative sentence hypotheses. For each of these hypotheses, we also determine the corresponding HMM phonetic segmentation. The phoneme sequence and its segmentation for each hypothesis is then rescored by the SSM system. The scores of the two systems are then combined, together with the grammar score, to determine the final answer. We developed a program that determines the linear combination of (log) scores which maximizes recognition accuracy.

One advantage of this method for combining different systems is that it is possible to integrate two radically different systems with a minimum of effort. Another important advantage is that the computation required for the second system is reduced tremendously, since it only needs to rescore those hypotheses that were judged as plausible by the first system. In fact, it is this second advantage that allows the SSM system to reduce its computational load tremendously and, thus, makes it into a computationally feasible approach.

The overall results of this cooperation were quite satisfying. First, the N-Best approach made it quite easy for the cooperation to take place, since we were able to send lists of hypotheses to BU and they were able to use them easily. Second, the computational requirements for the stochastic segment model were drastically reduced. Finally, the combination of the two recognition technologies was shown to result in a small but statistically significant improvement in recognition accuracy.

9.2 Combined HMMs and Neural Networks

Our research on the use of Neural Network techniques for speech recognition had two of the same problems that plagued the effort in Stochastic Segment Models. First, the computation required for training and recognition was prohibitive. Second, the accuracy of the new model by itself was not as good as the HMM alone. We applied the same approach—that of using the HMM as a preprocess to find the N-Best hypotheses—to the Neural Networks problem. One result was that the computational problems were largely avoided. The more important result was that we were able to demonstrate a significant advance in the state of the art in speech recognition by the appropriate combination of Neural Networks and HMMs[6].

9.3 Cooperation with Unisys

Since the research group at Unisys does not do speech recognition research of their own, they had the problem of how to apply their natural language research to the problem of spoken language understanding. They had initially considered the tightly coupled architecture formulated at MIT Lincoln Laboratory. However, this proved to be quite complicated and constraining. The N-Best Paradigm proved to be quite simple and effective. Specifically, at each of the evaluations of the ATIS systems, we have sent text of the N-Best sentence hypotheses output by the BYBLOS recognition system directly to Unisys (later Paramax). (These were the very same lists that were sent to the DELPHI system at BBN.) Unisys reported that it took only hours to configure their system to use this input, and that they found that lists of 10-20 sentences were quite sufficient for the purpose of understanding.

Chapter 10

Application in a Military Domain

In this chapter, we describe our work on DART, our demonstration military application, during the course of this project, and our creation of a videotape demonstrating the applications of spoken language system technology.

10.1 The DART Demonstration Application

The DARPA SLS Program is developing a technology that has been justified, at least in part, by its potential relevance to military applications. In an effort to demonstrate the relevance of SLS technology to real-world military applications, BBN undertook the task of providing a spoken language interface to DART (Dynamic Analytical Replanning Tool), a system for military logistical transportation planning.

We discuss the transportation planning process, describe the real-world DART system, identify parts of the system where spoken language can facilitate planning, and describe BBN's work of porting the HARC SLS system to the DART domain.

10.1.1 Transportation Planning

Logistical transportation planning is the process of determining how to get people and cargo from where they are to where they need to be. Inter-theatre movements of personnel and supplies around the world are currently planned for the Army, Navy, Air Force, and other services by USTRANSCOM (the US TRANSportation COMmand) which operates under the Joint Chiefs of Staff.

We chose TRANSCOM as an SLS application domain because it presented a number of advantages:

1. The application involves an essential military function and successful application of spoken language technology would be very desirable to DARPA's clients.
2. The concept of planning movements of people and supplies is understandable in both military and non-military contexts.
3. The application is non-trivial, and affords many opportunities for applying spoken language understanding. A series of demonstrations is possible, with various features at varying levels of effort.
4. Current efforts to improve the planning process using non-speech technology have been well-received, and cooperative users may be available as close as Scott Air Force Base near St. Louis.
5. An unclassified development database is available in Oracle on a Sun.

10.1.2 The DART System

BBN's DART (Dynamic Analytical Replanning Tool) project, sponsored by DARPA and RADC, demonstrated the operational impact of AI planning and scheduling technology on transportation planning at USTRANSCOM. DART addresses an urgent need for fast and accurate plan generation and evaluation to support both long-range, hypothetical planning and planning in such crisis-response operations as those in the Middle East.

The current DART system [40] is in use at Scott Air Force Base and other locations around the globe. The workstation environment which has been installed at TRANSCOM to support DART is already being used and has been credited with reducing routine plan analysis from 3 days to 1 days.

The architecture of the DART system is shown in Figure 10.1. The heart of the system is a relational database. The database is initialized with data from two sources, a database of transportation characteristics, and a Time Phased Force Deployment Database (TPFDD). TPFDDs are usually prepared in advance to deal with hypothetical military operations. In a crisis situation, the planner's task is usually to retrieve an applicable TPFDD, and to

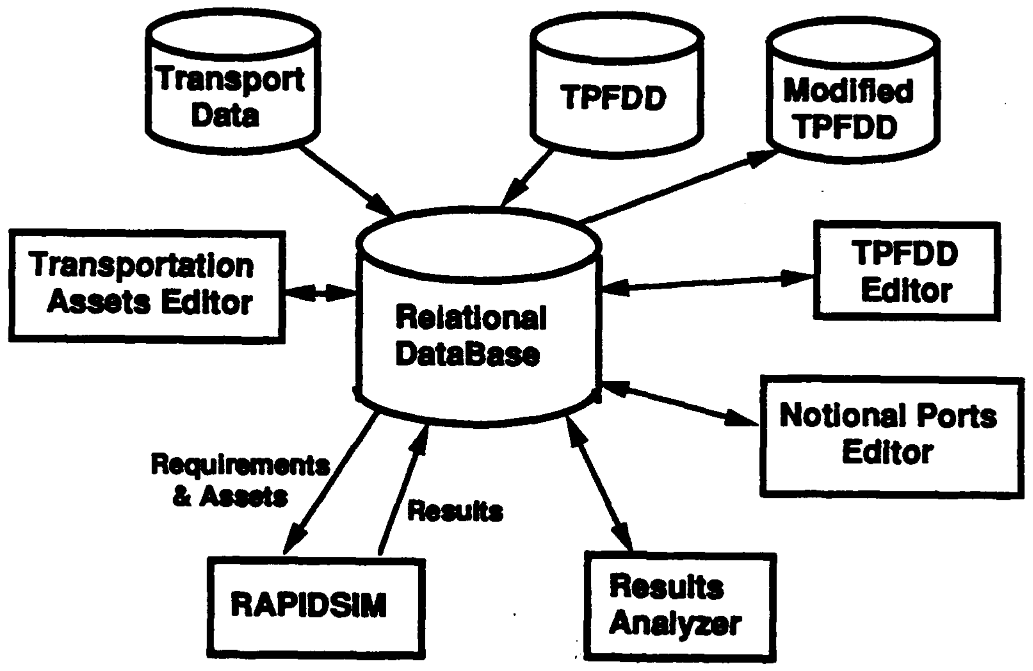


Figure 10.1: Architecture of DART

change it to fit that new situation. The output of the process is a modified TPFDD which can be used in subsequent planning and operational activities.

DART makes a number of tools available to the planner. These include a TPFDD editor for viewing units and making changes in their characteristics and transportation plans, a notional ports editor which allows ports to be combined for purposes of planning and simulation, a transportation assets editor which lets the planner modify the availability and characteristics of various transportation assets, the RAPIDSIM simulation system which can "run the current plan", and an analysis capability that enables the planner to examine the output of a RAPIDSIM run to determine whether or not the objectives were achieved

DART allows a planner to extract pieces of pre-planned movement records from a database by specifying simple constraints on up to five items: the units to be moved (a unit generally contains both personnel and cargo), the place of origin of the units, their port of embarkation, their port of debarkation, and their final destination.

The retrieved data is displayed in a spreadsheet-like window, horizontal bars showing the number of days each step of the transport is expected to take, with the color indicating whether the step is by land, sea, etc. An example of this window, and other parts of the normal DART display, is given in Figure 10.2.

10.1.3 The TPFDD Database

The database that underlies the entire planning process is called the TPFDD (Time Phased Force Deployment Database) [51]. The TPFDD development database that is unclassified and available in Oracle has 50–100 MB of data, in 13 tables and about 500 fields. This data represents approximately 20,000 cargo movement records, 9,000 unit movement records, and a smaller number of personnel movement records. Each record contains, among other information:

- location of origin
- POE (port of embarkation)
- intermediate locations, if any
- transportation mode (land, sea, air)
- transportation provider
- POD (port of debarkation/discharge)
- location of destination
- RLD (ready to load date) at origin
- ALD (available to load date) at POE
- EAD (earliest arrival date') at POD
- LAD (latest arrival date) at POD
- RDD (required delivery date) at destination

10.1.4 DART plus SLS

Natural language access (both spoken and typed) increases the utility of the DART interface by providing capabilities that are not available in the non-language interface, and it can decrease the task completion time for operations that can be expressed more concisely in words than in mouse actions.

We identified six areas of the DART system where natural language will provide increased functionality for this military system:

1. the TPFDD editor, which allows users to create and modify entries in the Timed Phased Force Deployment Databases that specify movement requirements for the personnel and materiel involved in planned military operations,
2. the transportation assets editor, which is used to view and change the number and type of transportation assets (ships, planes, etc.) and the days when they are available,
3. the notional ports editor, which is used to combine actual ports (sea ports, air ports, or other geographical locations) into single "notional" ports to simplify subsequent simulations of planned movements,
4. the analysis of results from the RAPIDSIM simulation of the current plan's execution,
5. universal (that is, available throughout the whole DART system) access to information in the TPFDD database that underlies the planning system,
6. menu navigation through the DART system, so that a user can use a single verbal command instead of a lengthy sequence of mouse (and possibly keyboard) operations.

Each of these opportunities for adding spoken language to the DART interface has separate pros and cons. They vary in expected vocabulary size, likely language complexity, ease of interface to DART, and utility for the user.

For example, in the notional ports editor, the user is likely to want to give short commands to the system ("Show me Travis Air Force Base", "Zoom in around Charleston", "What's this port?", "Show the nearest military airport", "Compute the notional port assignments"). The planner is also likely to refer only to the geographical locations that are displayed on the current map, which reduces the vocabulary (and the perplexity) considerably.

Universal database query, on the other hand, will involve complex language ("What percentage of the Navy units headed for air force bases in Tunisia that are available to load from US ports prior to day 20 contain hazardous cargo?"). This part of the application will also require a very large vocabulary, since virtually any geographic location or other word from the database can be used in a query. We estimate that even for just a good demonstration, the vocabulary will need to be about 5000 words.

10.1.5 Current Status

By the end of this project, we transferred from the small in-core planning database that we developed for demonstrating HARC to using the real TRANSCOM development database in

Oracle. We developed an initial interface between the DART user interface and the windows indicating activities in speech processing and NL understanding. We have implemented a mechanism to allow units that are retrieved via a natural language query to be imported into the DART plan display.

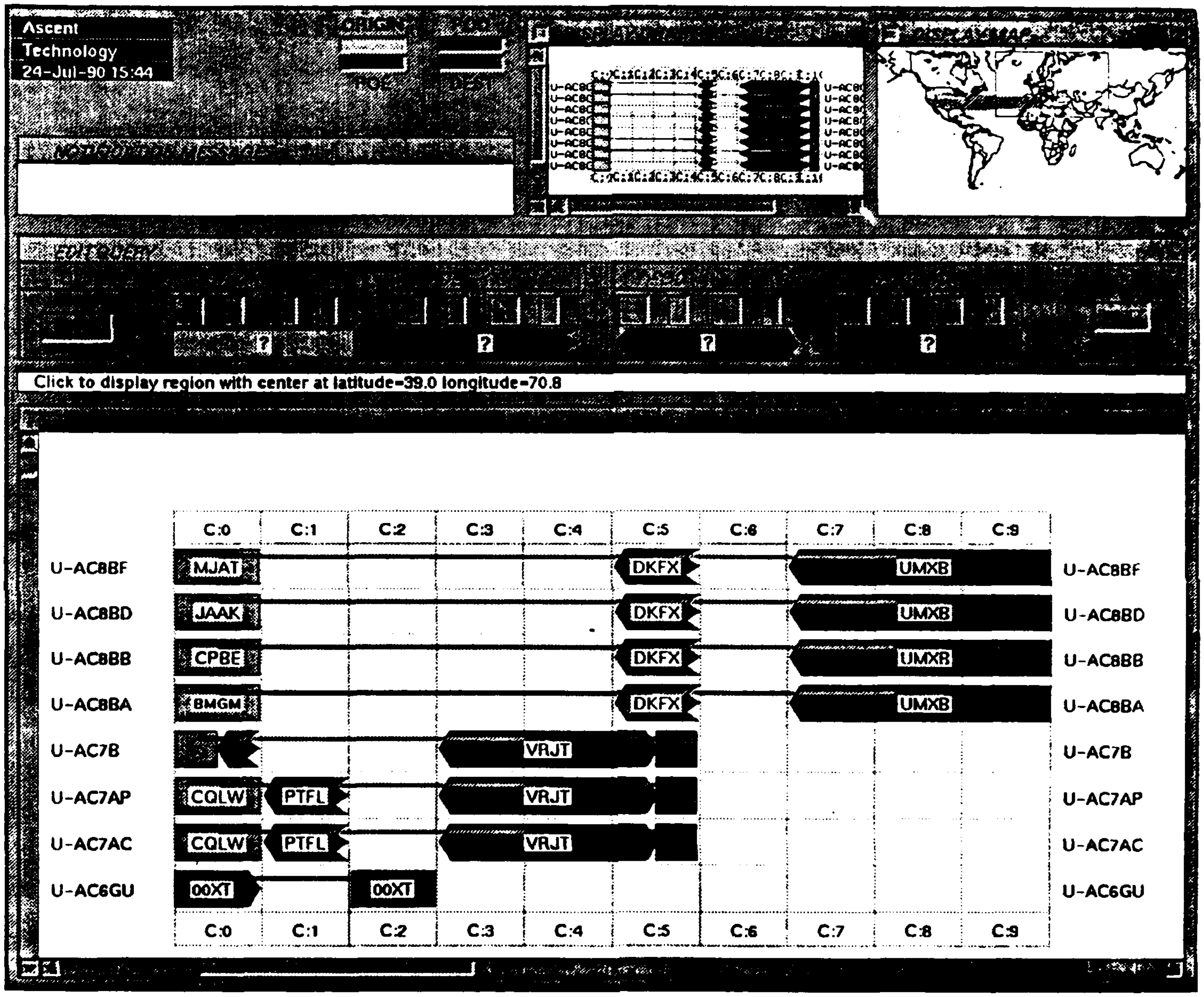
10.2 Videotape

To demonstrate our various capabilities in real-time spoken language systems, we developed a videotape that included demonstrations of:

- The real-time N-best BYBLOS system working in speaker-independent mode in the resource management domain.
- Speaker adaptation to improve recognition performance for non-native speakers of American English.
- The real-time HARC spoken language system in the military logistical planning domain, showing the combination of speech recognition and natural language understanding.
- The DART application and the DART+SLS demonstration of the use of spoken language to enhance system capabilities in one of the six areas described in the previous section.

The DART+SLS demo describes the task of TRANSCOM planners and shows examples of the interactions that are possible with spoken language to facilitate the planner's work. For example, because the DART+SLS system has independent access to the system's database, the user can ask questions about the data which would have been difficult or impossible to answer in the existing DART system. Also, certain queries that currently require a large number of mouse clicks can be asked with a single spoken utterance.

This videotape was presented at the February, 1991 DARPA Speech and Natural Language Workshop.



Click to display region with center at latitude=39.0 longitude=70.8

Figure 10.2: The DART User Interface

Chapter 11

Common Data Collection and Evaluation

11.1 Development of a Performance Evaluation Methodology

While the Continuous Speech Recognition (CSR) community has had an established methodology for performance evaluation on a common corpus for several years, there was no similar methodology for the evaluation of natural language systems or of spoken language systems at the beginning of this contract. BBN played an important role in the development of a common evaluation methodology. BBN detailed the basic methodology used for NL and SLS evaluation in [26]. This work specified the following aspects of evaluation which are still in use today:

- Evaluation utilizes input-output pairs.
- Input is a spoken or written utterance.
- Output is the answer retrieved by the input from a common database.
- The correct answer is retrieved from the common database.
- Both the reference answer and the hypothesis answer to be evaluated are presented in a common answer specification (CAS) form.
- An automatic "comparator" is used to compare the answers and score the system output, allowing for some unimportant differences in presentation of the data (such as order of elements) to be disregarded.

BBN's proposal [26] also detailed the requirements for an automatic software comparator which would match a hypothesis CAS produced by an NL or SLS system under evaluation with a reference CAS. (Among such requirements are the necessity for a epsilon factor for numeric comparison of floating point numbers.) These proposals were finally adopted by the DARPA community with little modification. BBN also produced the first comparator program in LISP, which was made available to the SLS community. The C-based comparator produced by NIST eventually recapitulated the functionality of this system. For sites which do not possess Oracle databases, BBN also documented its Common LISP-based data base and its retrieval language, ERL, [77], and sent the documentation and a no-cost license agreement for the data base interface to CMU, Dragon Systems, MIT, NIST, SRI International, TI, and Unisys, thereby making the database software available to the general DARPA SLS community.

We also implemented an early Wizard of Oz System [26], documented it, and delivered it to Texas Instruments so that they could gather realistic speech input to a natural language interface to a database. We trained several TI employees to serve as Wizards.

We have continued our strong participation in developing a methodology for common evaluation of spoken language systems, especially the evaluation of natural language understanding systems [15], [12], [14] For example, we made our database expertise available to TI and helped them in their effort to produce a relational ATIS database from the original ATIS data obtained from OAG. We also helped in specifying various aspects of the Wizard data collection scenario and the performance evaluation process with NIST, including specification of general templates for descriptions of the CSR and NL systems submitted for evaluation. Examples of these templates, describing DELPHI, BYBLOS, and HARC, appear in Section 11.6

11.2 ATIS Data Collection

During the summer of 1991, we developed a Wizard system for collecting data in the ATIS domain, collected data in the formats agreed upon by the MADCOW committee, and sent the results to NIST. We first obtained the new database (rdb3-beta), a modified version of the original ATIS database, and installed this on our machines. Since this database contained new tables, fields, and field values, we next modified the interface between DELPHI and the database accordingly. We then integrated the modified DELPHI system into our Wizard set-up for collecting data. As part of our data collection effort, we defined 24 scenarios, some of which we obtained from MIT, TI, and SRI. These scenarios are presented in Section 11.3 at the end of this chapter. We also collaborated with SRI in defining a common scenario for cross-site data collection.

For our Wizard data collection, we produced a "Wizard" data collection setup to elicit speech from subjects by presenting them with a computer system that appeared to understand them (that is, the computer presented responses to what the subject said), but which actually had a human "wizard" listening to the spoken questions and typing the commands to the system to produce the answer.

This data collection setup employed a novel, interactive subject and wizard interface based on X-windows. The subject's queries and answers were stacked on the color screen for later examination or other manipulation by the subject. The system also used our real-time BYBLOS speech recognition system as the front end. Real-time speech recognition was used primarily to discourage subjects from speaking too sloppily (the wizard had the choice of using the speech recognition output or correcting it).

The scenarios that were used to elicit speech in the data collection setup included both trip planning scenarios and problem solving scenarios that involved more general kinds of database access. We believe that this data provided a richer range of training language than trip planning alone, and thus will help to distinguish systems that are unnecessarily narrow in focus from those with more general capabilities. The Wizard set-up used the following protocol:

- The Wizard and Subject are seated at separate Sun workstations, each equipped with an X-windows display and a mouse.
- The Wizard tells the Subject about the data collection, explains the display and how to manipulate it, gives the user written instructions and a list of scenarios, and goes through a practice scenario to familiarize the subject with the process.
- Using the mouse and a "control-panel" icon, the Subject specifies the scenario in effect for the current session.
- During the course of the session, the Subject precedes each utterance by clicking on a software push-to-talk button with the mouse, to activate the speech system.
- The Subject speaks into a close-talking microphone, which sends the speech to our speech recognition system, BYBLOS. The speech is digitized and stored in a `.wav` file.
- BYBLOS produces its N-best hypotheses about the input utterance, which are displayed to the Wizard.
- The Wizard chooses one of these, possibly editing it so that it is a reasonable quick transcription of what the speaker said, and displays it to the Subject.
- DELPHI processes the query.

- When the system produces an answer, this is displayed to the Subject. If DELPHI fails to process the utterance, the Wizard either displays an error message to the Subject, or submits a modified query to produce an answer.
- This continues until the end of the scenario, which is signalled by the Subject clicking on an "End Scenario" button with the mouse.

Each session produced a separate logfile, in an internal format. Each logfile was processed by a LISP program to transform it into a form consistent with the MADCOW specifications for common logfiles. We also manually produced the `.sro` transcription files by listening to each utterance. We obtained a program from NIST for checking conformance of our digitized speech (`.wav`) files with NIST's standard, to ensure their correctness before shipping them to NIST.

We used this data collection system to collect data from 62 subjects, 6 of whom came back for a second session. We collected over 2200 sentences in all and delivered them to NIST, fulfilling the cross-site agreement to collect this amount of data by the end of August. BBN, in fact, was the only site to collect the targeted number of acceptable queries by the original deadline of 1 September, 1991. We present a breakdown of the data by gender and by scenarios in Table 11.1. In addition, we submitted to NIST, as required, the text of all scenarios as well as the instructions given to subjects. Our scenarios are presented in Section 11.3 and the subject instruction in Section 11.4. We also include an actual sample session in Section 11.5.

11.3 BBN ATIS Scenarios

This section contains all the scenarios used for data collection by BBN during the summer of 1991, grouped by scenario type.

11.3.1 Initial scenario

In order to accustom the user to the push-to-talk button and adjust volume levels, the first query each subject asked was the following.

P1: What is the name of the airport in Boston?

Speakers

Male	34	1209 utterances	35.6 utterances/speaker
Female	8	1068 utterances	38.1 utterances/speaker
Total	62	2277 utterances	36.7 utterances/speaker
Average	4.95 scenarios/speaker		

Scenarios

A = flight planning scenarios (includes the common scenario used by all sites)
B = problem solving scenarios
C = short planning scenarios (mostly the same as some of MIT's scenarios)

	#scenarios	#utterances	#utterances/scenarios
A	108	994	9.20
B	86	705	8.20
C	113	578	5.12
Total	307	2277	7.51

Common Scenario: 33 (10.75% of total scenarios)

Table 11.1: Breakdown of BBN Wizard Data by Gender and Scenario

11.3.2 Practice Scenarios

These scenarios could be answered in one or two questions and were designed to acquaint the user with the type of information in the database and with interacting with the system.

C1: Find the cheapest (or the most expensive) one-way fare from one city to another.

C2: Find the earliest (or latest) flight from one city to another that serves a meal of your choice.

C3: Determine the type of aircraft used on a flight from one city to another that leaves before (or after) a certain time of the day.

C4: Find a transcontinental (east coast to west coast) flight on your favorite airline from one city another that makes a stop-over in a city of your choice.

C5: Determine the longest day trip you can make between two cities of your choice. (You are trying to maximize the amount of time on the ground at your destination)

C6: Find a flight between two cities. The flight should leave in the afternoon and arrive around your dinner time. It should be a non-stop flight.

11.3.3 Trip Planning Scenarios

These are scenarios that required longer and more complicated interactions with the system. Each scenario involves planning a trip.

A1: Plan the travel arrangements for a small family reunion. The reunion will be held in Baltimore. Three family members will be attending the meeting:

One of them, who typifies the "high class" life style, will be coming from Denver; find first-class travel arrangements for her on United Airlines.

The second has a life style described as "economy"; find travel arrangements from Dallas to Baltimore for the least amount of money possible.

The third has an "adventurous" life style (he loves to hang glide and fly in small airplanes); he lives in Pittsburgh. To make this person's trip enjoyable, find a flight from Pittsburgh to Baltimore on a plane that can hold the smallest number of passengers.

You will also need to specify the date of arrival for each of these family members.

A2: You live in Philadelphia. You need to make a business trip to San Francisco next week. You have an old friend in Dallas and you'd therefore like to spend the afternoon in Dallas on your way out to San Francisco. You'd prefer to fly first class on American. Find out what kind of aircraft you'll be flying on.

A3: Choose two cities: city A ____________ and city B ____________.

You live in city A. You want to combine a business trip to city B with pleasure by taking your spouse along, but you don't want to have to pay for your spouse's travel. Fortunately, the company will reimburse you for an amount equal to the price of a first-class ticket. Plan a trip that will allow you to stretch the travel allowance to cover the expenses for both you and your spouse.

A4: Choose three cities: A ____________, B ____________ and C ____________. You live in city A. You have only three days for job hunting and have arranged two job interviews in cities B and C, each lasting at least three hours. Pick the cities and plan the flight itinerary.

A5: Choose city A ____________, your starting point, two other destination cities (city B ____________ city C ____________), and then solve the following scenario.

You have only three days for job hunting, and you have arranged job interview in two different cities. (The interview times will depend on your flight schedule.) Start from city A and plan the flight and ground transportation from city B and city C, and back home to city A.

Scenario A5 was the common scenario specified by BBN and SRI.

11.3.4 Data Base Exploration Scenarios

These scenarios also involve longer and more involved interactions with the system. Unlike the previous scenarios, these encourage the user to treat the ATIS system as a data base for free exploration. We included these scenarios to produce natural language different from that used for trip planning.

B5: You have heard in a commercial that one airline claims to have more flights with some particular class (you can't remember whether it is business class, or first class, or some other class) than all other airlines combined. You also forgot which airline made that claim. Find out which airline, if any, can reasonably make such a claim, and which class the claim was about.

B6: An airline has a "hub" in a city if it has a lot more flights into and out of that city than other airlines do. Determine whether any of the airlines represented in the database have hubs, and, if so, which cities are hubs for which airlines. (You may stop with one airline and one hub, if you wish.)

B7: Pick an airline and pretend that you are in charge of suggesting new routes and schedule changes for that airline. Investigate the database to determine whether there is a gap that your airline could fill. A gap could be a route between two cities that you don't currently serve, or it could be a part of the day that isn't covered by you (or your competition) for travel between cities you already serve.

B8: Two airlines might be suspected of being in collusion if they appear to have divided up territory (or time of day in a single city) so as not to compete with one another, or if they have nearly identical prices for the same service. Pick any two airlines, and determine whether they might be colluding.

B9: Choose a city ____________ and a period of time during the day ____________. Suppose that the airport in this city has closed for that period of time due to minor construction work. Of the airlines that use this airport, find the airline that will be least affected by the airport closing. Also, find out what is the maximum possible number of passengers that might be affected (based on the seating capacity of the types of aircraft used in those flights).

B10: Choose an airport: ____________. The "congestion period" in an airport is that 3-hour period in which there are more take-offs and landings than in any other 3-hour period of the day. For the airport you have chosen, determine its congestion period.

B11: Choose one city from each of the following pairs of cities: Boston/Washington DC; Denver/San Francisco. Find an airline which provides only direct flights between the two chosen cities. Also, is there an airline that provides only connecting flights between the same two cities? If so, determine the types of aircraft used on those flights.

B12: Choose one of the following cities: Atlanta, Denver, Dallas. Suppose that due to bad weather the airport in this city will be closed for the morning. Find all the flights that make a connection in this airport and that will be affected. Find other similar (same origin and final destination) afternoon flights that connect through the same airport.

B13: Choose two cities A ____________ and B ____________ Choose any three airlines that fly from city A to city B.

You have always been curious as to the price differences among the various airlines. Perform some research on price comparisons for the three selected airlines, studying flights from city A to city B with similar characteristics (same class of service, similar departure time, etc). In your opinion, which airline is the most "greedy" and which is the least "greedy" among the three airlines.

B14: Choose an airport: ____________. An airline dominates a time slot in an airport if that airline has more takeoffs and landings in that airport during that time slot than any other airline. A business time slot is early morning or evening. Does any airline dominate the business time slots?

B15: An airline is "dependent" on a manufacturer if most of the flights on that airline use planes from that manufacturer. Are any airlines "dependent" on a manufacturer?

B16: Which types of aircraft are used for transcontinental (east coast to west coast) flights? Are any of these types of aircraft used for flights which are not legs of transcontinental flights?

B17: An airline cuts cost on a route if it serves only a snack to coach passengers when it serves a full meal to first class passengers. Pick two airlines and find out if there is a route on which they cut cost in that manner.

11.4 BBN Subject Instructions

ATIS Data Collection
Instructions to Subjects

Thank you for participating in our data collection project.

1. Getting Acquainted

What is ATIS?

ATIS, or Airline Travel Information System, is a voice-operated system under development at BBN. The system is designed to give answers to spoken queries about flight information. The flight information that the system knows about is a subset of the information contained in the Official Airline Guide (OAG). (See Section 2 entitled "What kind of information is in the ATIS system".)

What is the purpose of this data collection effort?

Because the system we have is still under development, it will be able to answer many (we hope) but not all of your queries. The purpose of this data collection is to give us a larger sample of queries from actual users who are trying to use the system. We plan to use the collected data to help us improve the performance of the system.

What am I supposed to do?

Your task is to use the ATIS system to help you in solving several flight-related scenarios given to you by the data collector at the time of data collection. In the process of solving the scenario, don't make a special effort to restrict yourself to ask queries that are the most relevant to that scenario. In particular, when you are just starting a scenario you may want to explore the information that ATIS contains.

How to use the system

See Section 3 for a complete set of instructions on how to use the system. In addition, you will receive instructions on the use of the system from the data collector. The data collector will be monitoring the whole data collection session.

Briefly, you request information from ATIS by using the microphone to talk to the system. The screen will show what the system thinks you said and what the answer to your query is.

Remember: *The purpose of this process is to collect speech data; the more queries you ask the better. Don't try to think of the shortest way you can get your information. Don't be afraid to explore.*

2. What Kind of Information is in the ATIS System?

ATIS knows about flights between a limited number of cities (see list below). For each flight, ATIS knows about the following:

Airline name (American, Continental, Delta, TWA, Eastern, Lufthansa, Midway, United, and USAir)
Airline abbreviation (AA, CO, DL, TW, EA, LH, ML, UA and US).
Flight number
Originating city
Destination city
Departure and arrival times (also elapsed time of a flight)
Dates (July 25, 7th of November, etc.)
Days of service (Sunday, Monday, etc.)
Classes of service (first class, business, etc.)

Number of stops
Fares (these vary depending on flight, day, class of service, and whether the fare is one way or round trip)
Meals
Ground transportation between airport and downtown, and its cost
Type of aircraft (jet, turboprop, DC10, 72S, 727, etc.)
Seating capacity of aircraft

You may ask the system questions about any abbreviations you don't understand.

Cities and Airports that ATIS knows about:

CITY/AIRPORT	AIRPORT CODE	ALTERNATE AIRPORT NAME
Atlanta	ATL	William B. Hartsfield
Baltimore/Washington	BWI	
Boston	BOS	Logan
Dallas/Fort Worth	DFW	
Denver	DEN	Stapleton
Oakland	OAK	
Philadelphia	PHL	
Pittsburgh	PIT	
San Francisco	SFO	

There are 11 cities served by 9 airports. Baltimore and Washington are served by the same airport; likewise for Dallas and Fort Worth. The airports in Oakland and San Francisco serve both cities.

3. How to Use the ATIS System

3.1 Scenarios

After an initial training period, the data collector will give you several scenarios to solve using the ATIS system. Below is the procedure to follow for each of the scenarios:

• **Begin Scenario:** To begin a new scenario, use the mouse to click (with the left-most mouse button) the **Begin Scenario** button in the top-left of the ATIS screen (see Figure 1). A menu of scenario numbers will pop up. Click the identification number of the scenario that you will be working on. (The Begin Scenario button will now say End Scenario; see below.)

Note: Make sure you write down the specifics of the problem you have selected to solve. This may avoid getting you and the system confused! Use the notepad provided to you by the data collection manager to write down notes for yourself as you wish throughout the data collection session

• Query Sequence: You solve the scenario by asking the ATIS system a sequence of queries. To ask a new query:

- Click **New Query** button in the top-left corner of the ATIS screen. The System Status display will show the word "Listening", when the system is ready to accept speech input.
- Speak your query into the microphone.
- After a short wait, the system will show on a new card what it thought you said (see Figure 1).
- After another (usually longer) wait, the system displays on the card an answer to your query. (Sometimes, this wait can be quite long; please be patient.)

Repeat the above sequence for every new query until you feel you have finished working with that scenario. Solving a scenario means that you feel you have obtained from ATIS the flight information specified in the scenario.

If after saying a query you decide that you wish to abort that query (e.g., you made a false start or you changed your mind about what you wanted to say), simply click on the **Abort Query** button.

• **End Scenario:** To end a scenario, click the End Scenario button on the ATIS screen. Now, you can begin another scenario by following the above procedure.

3.2 ATIS Screen

Figure 1 shows the basic ATIS screen, which you will use to control the session and the system uses to communicate with you. The ATIS screen has two main displays: The **top level menu** and the **cards display**.

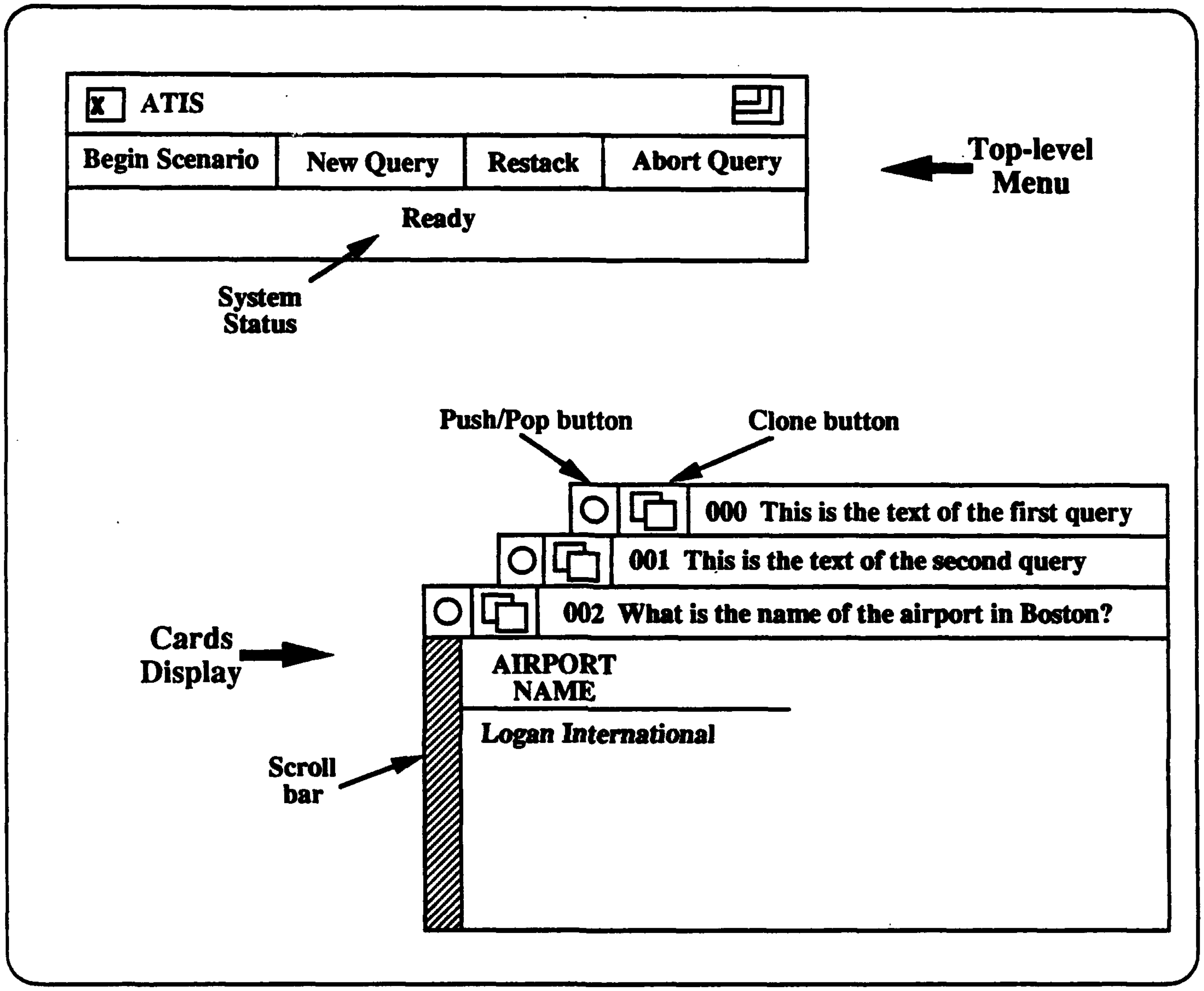


Figure 1. ATIS Screen

Begin Scenario	- Pop the scenarios menu
New Query	- Start recording a new query
Restack	- Return all cards to their original position
Abort Query	- Abort the query
⧉ Clone Button:	- Clone (copy) a card (click with left mouse button and drag to desired spot on the screen)
○ Push/Pop Button:	- Pop a hidden card to the top of the card stack; push a card back to its original position.

• **Top Level Menu:** This menu is shown in the top-left part of the ATIS screen. It contains a number of buttons and status windows. We have already described how several of the buttons and status windows are used.

• **Cards Display:** The cards display contains your queries and their answers. Each card corresponds to a single query. The top line of each card shows the card number and what the system thought you said. In the main body of the card, the system displays the answer to the query. As you say the different queries, the cards appear automatically and stack themselves in the lower right part of the screen. The stacking is done such that the old queries are visible but not their answers; only the most recent card is fully visible. After about ten cards have been created, the oldest one will be deleted automatically whenever a new one is created.

To the left of each card is a **scroll bar** that allows you to scroll that card vertically in case the answer occupies more space than allotted.

You will often find the need to look back at the answers to some of the previous queries. The answers to older queries can be seen by simply clicking on the **push/pop button** in the top-left corner of the card of interest. Clicking that button will bring forward (pop) that card and you will be able to see the whole card. Clicking the same button again will push that card back where it came from. You can pop a number of cards in that manner. To avoid having to push them all back by clicking the push/pop button, simply click on the **Restack** button in the top-level menu.

Note: The system only remembers the ten most recent queries. Cards that disappear from the screen can never be retrieved.

3.3 Clone Cards

In addition to being able to look back at previous queries and their answers, it is often convenient to be able to make a copy of a card for later reference. This is particularly important if you suspect that the card of interest corresponds to an old query that soon may disappear from the screen.

To make a copy of a card, you clone it. To do that, click the **clone button** in the top-left part of the card and drag the card-skeleton that appears to the desired spot on the screen, then click the mouse again. Figure 2 shows a card that has been cloned. A clone card looks the same as a card in the Cards Display except that it has an additional top bar and different buttons. Unlike the cards in the Cards Display, the clone card can be resized using the **Resize button** in the top-right corner of the card.

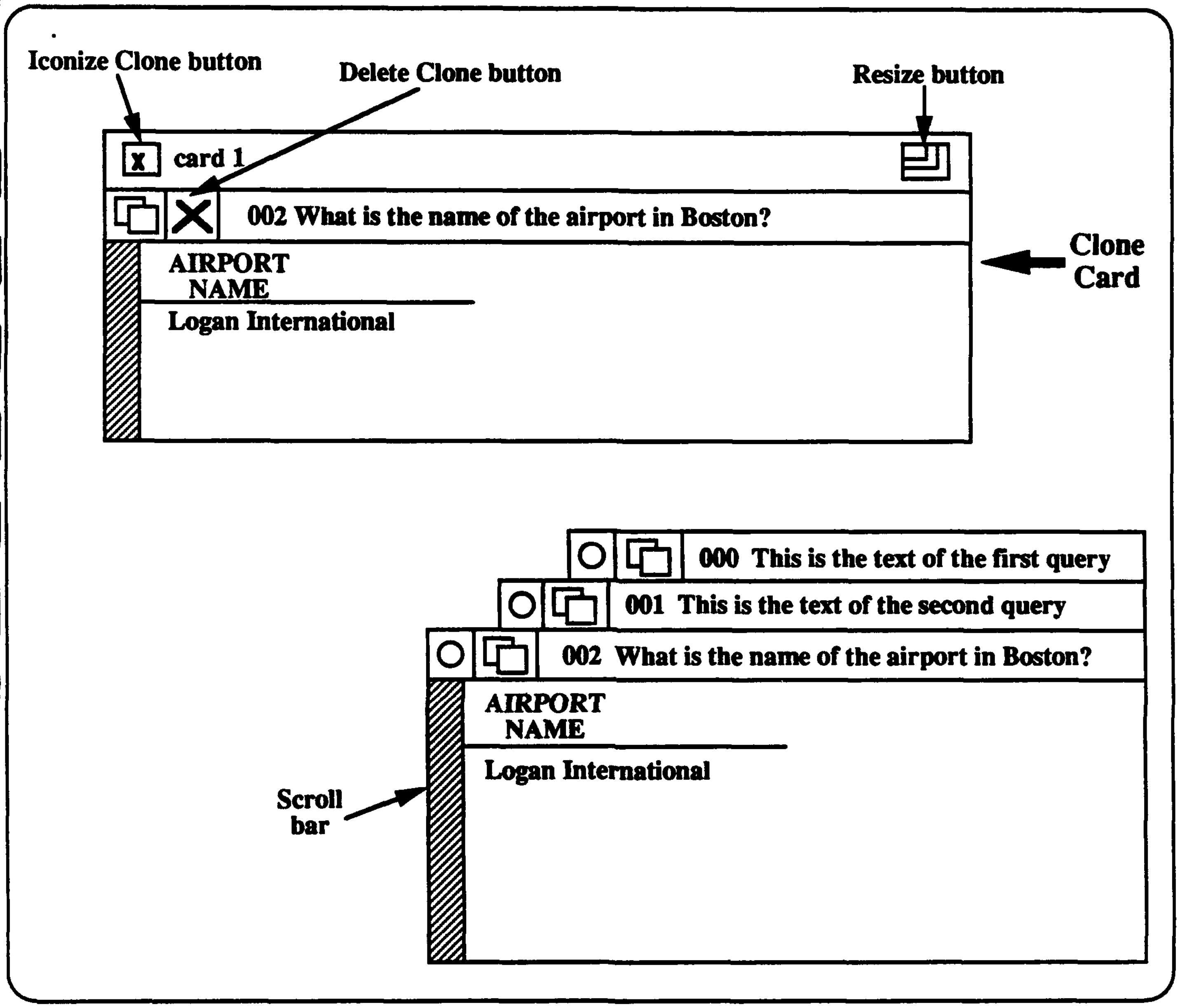

Iconize Clone button
Delete Clone button
Resize button
card 1
002 What is the name of the airport in Boston?
AIRPORT NAME
Logan International
Clone Card
000 This is the text of the first query
001 This is the text of the second query
002 What is the name of the airport in Boston?
AIRPORT NAME
Logan International
Scroll bar

Figure 2. Clone Card

Resize Button:	- Resize card (e.g., make it wider to see more of the answer).
Iconize Clone Button:	- Iconize a clone card; to reopen the icon click on the appropriate icon (e.g., card 1)
Delete Clone Button:	- Delete clone card
▨ Scroll Bar:	- Scroll card up/down (e.g.,to see other parts of the answer)

The clone card can be deleted from the screen by clicking the **Delete Clone** button in the top-left corner of the card. A clone of the same card can be made again from the Cards Display, but only if that card is still visible in the Cards Display. To make the clone card disappear temporarily, click on the **Iconize Clone** button in the left part of the query line of the clone card. The clone card will disappear but an icon corresponding to that card will appear in the top right part of the screen with the name of the card in it. The clone card can be made visible again by clicking in the icon for that card.

You may clone as many cards as you like, within the limits of the size of the screen.

3.4 Messages from the system

If the system is unable to understand or answer your request, you will receive a brief message on the screen. Read the message and click in the box under it that says "OK" when you are ready to continue.

4. Tips for Success

Remember, this is an experimental system. It can be quite slow at times and it may not be able to answer all your queries. **Please be patient.** Here are some tips on how to make the session go smoothly.

Be specific. The system tends to take what you say quite literally.

Don't be long winded. Many short queries generally work better than few long ones.

Don't pause. Short pauses are OK, but if you pause too long in the middle of a question, the system will think that you have finished speaking, and will stop listening to you.

Think before you talk. Compose your query, press the New Query button on the screen, and begin talking soon after the system status changes from "ready" to "listening".

Be natural. Don't try to speak "computerese", just ask the questions in ordinary English and using your natural voice.

Refer to notes. You might want to keep this document turned to the page titled "What kind of information is in the ATIS System?" to see the kinds of things you can ask about.

Ask it! If you're not sure whether to ask a particular question or not, go ahead and give it a try!

Don't get overwhelmed when receiving large amounts of information from ATIS. When ATIS responds with an answer that contains too much information, you may want to ask additional queries that will constrain that information.

11.5 A Sample BBN Session

As an example of the types of interactions users had with our ATIS system, we include here the record of a user's interaction with the system to solve a particular scenario. We include the natural language queries alone in Section 11.5.2.

11.5.1 The Session

```
[RDBVersion:] rdb3

[Timestamp: BBN Started log at 16:01:05]
[Speaker:] e3
[Date:] 050691
[ScenarioType:] BBNA4

[Begin ScenarioDescr:]
A4: You live in city A. You want to combine a business trip
to city B with pleasure by taking your spouse along. You will
receive a fixed travel allowance, based on a first-class
ticket. Plan a trip that will allow you to stretch the travel
allowance to cover the expenses for both you and your spouse.
[End ScenarioDescr:]
[UtteranceID:] 001
[Timestamp: Sent speech for utterance 001 at 16:01:52]
[Timestamp: BBN sent transcription for utterance 001 at 16:02:05]

[Begin Utterance: 001]
WHICH AIRLINES FLY FROM BOSTON TO SAN FRANCISCO
[End Utterance: 001]

[Timestamp: Sent answer for utterance 001 at 16:02:28]
```

[Begin Result: 001]

```
AIR  AIRLINE                 AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV
LINE NAME                                                  TIME  TIME
---- ---------------------- -------------------- ---- --- ----- -----
US   USAIR                   US81/US37            BOS  SFO 12:10 16:52
UA   UNITED AIRLINES         UA343                BOS  SFO 08:40 13:36
UA   UNITED AIRLINES         UA343/UA551          BOS  OAK 08:40 13:39
DL   DELTA AIR LINES, INC.   DL975/DL99           BOS  SFO 15:20 21:15
UA   UNITED AIRLINES         UA201/UA343          BOS  SFO 08:38 13:36
UA   UNITED AIRLINES         UA281/UA673          BOS  OAK 17:20 22:05
DL   DELTA AIR LINES, INC.   DL831/DL149          BOS  SFO 08:05 13:25
UA   UNITED AIRLINES         UA281/UA297          BOS  SFO 17:20 22:14
DL   DELTA AIR LINES, INC.   DL487/DL395          BOS  SFO 18:45 23:50
UA   UNITED AIRLINES         UA201/UA551          BOS  OAK 08:38 13:39
UA   UNITED AIRLINES         UA93                 BOS  SFO 17:45 21:27
TW   TRANS WORLD AIRLINE...  TW61                 BOS  SFO 18:10 21:37
CO   CONTINENTAL AIRLINES    CO1765               BOS  SFO 17:40 23:10
AA   AMERICAN AIRLINES       AA813                BOS  SFO 11:49 17:58
DL   DELTA AIR LINES, INC.   DL169/DL887          BOS  SFO 11:32 16:50
UA   UNITED AIRLINES         UA21                 BOS  SFO 08:00 11:31
```

[End Result: 001]

[UtteranceID:] 002
[Timestamp: Sent speech for utterance 002 at 16:02:53]
[Timestamp: BBN sent transcription for utterance 002 at 16:03:08]

[Begin Utterance: 002]
WHICH OF THOSE FLIGHTS ARE FIRST CLASS SERVICE
[End Utterance: 002]

[Timestamp: Sent answer for utterance 002 at 16:03:58]
[Begin Result: 002]

```
AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT    FARE
                              TIME  TIME  CODES          DAYS      CODE
-------------------- ---- --- ----- ----- -------- ----- --------- -----
UA21                 BOS  SFO 08:00 11:31 D10      B     DAILY     F
UA343/UA551          BOS  OAK 08:40 13:39 D8S/727  BL    SU        F
DL975/DL99           BOS  SFO 15:20 21:15 757/767  SD    DAILY     F
```

```
UA93                 BOS  SFO 17:45 21:27 D8S      D     DAILY     F
DL831/DL149          BOS  SFO 08:05 13:25 757/767  BL    DAILY     F
UA201/UA551          BOS  OAK 08:38 13:39 D8S/727  BL    NOT SU    F
CO1765               BOS  SFO 17:40 23:10 M80      D     NOT SA    F
UA281/UA297          BOS  SFO 17:20 22:14 D8S/72S  D     DAILY     F
UA343                BOS  SFO 08:40 13:36 D8S      BL    SU        F
DL487/DL395          BOS  SFO 18:45 23:50 72S/757  DS    DAILY     F
UA281/UA673          BOS  OAK 17:20 22:05 D8S/733  D     DAILY     F
TW61                 BOS  SFO 18:10 21:37 L10      D     DAILY     F
US81/US37            BOS  SFO 12:10 16:52 M80/733  SL    DAILY     F
UA201/UA343          BOS  SFO 08:38 13:36 D8S/D8S  BL    NOT SU    F
AA813                BOS  SFO 11:49 17:58 767      LD    DAILY     F
DL169/DL887          BOS  SFO 11:32 16:50 757/757  LS    DAILY     F
```

[End Result: 002]

[UtteranceID:] 003
[Timestamp: Sent speech for utterance 003 at 16:05:22]
[Timestamp: BBN sent transcription for utterance 003 at 16:06:07]

[Begin Utterance: 003]
LIST THOSE FLIGHTS LEAVING BEFORE 9 AM DAILY
[End Utterance: 003]

[Timestamp: Sent answer for utterance 003 at 16:07:40]
[Begin Result: 003]

```
AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT
                              TIME  TIME  CODES          DAYS
-------------------- ---- --- ----- ----- -------- ----- ---------
UA21                 BOS  SFO 08:00 11:31 D10      B     DAILY
UA343/UA551          BOS  OAK 08:40 13:39 D8S/727  BL    SU
DL831/DL149          BOS  SFO 08:05 13:25 757/767  BL    DAILY
UA201/UA551          BOS  OAK 08:38 13:39 D8S/727  BL    NOT SU
UA343                BOS  SFO 08:40 13:36 D8S      BL    SU
UA201/UA343          BOS  SFO 08:38 13:36 D8S/D8S  BL    NOT SU
```

[End Result: 003]

[UtteranceID:] 004
[Timestamp: Sent speech for utterance 004 at 16:08:26]
[Timestamp: BBN sent transcription for utterance 004 at 16:08:41]

[Begin Utterance: 004]
WHICH OF THOSE FLIGHTS ARE NONSTOP
[End Utterance: 004]

[Timestamp: Sent answer for utterance 004 at 16:09:13]
[Begin Result: 004]

```
AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT
                              TIME  TIME  CODES          DAYS
-------------------- ---- --- ----- ----- -------- ----- ---------
UA21                 BOS  SFO 08:00 11:31 D10      B     DAILY
```

[End Result: 004]

[UtteranceID:] 005
[Timestamp: Sent speech for utterance 005 at 16:10:12]
[Timestamp: BBN sent transcription for utterance 005 at 16:10:28]

[Begin Utterance: 005]
LIST ALL FLIGHTS ON UNITED FROM SAN FRANCISCO TO BOSTON
[End Utterance: 005]

[Timestamp: Sent answer for utterance 005 at 16:10:36]
[Begin Result: 005]

```
AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT
                              TIME  TIME  CODES          DAYS
-------------------- ---- --- ----- ----- -------- ----- ---------
UA20                 SFO  BOS 13:40 22:02 D10      L     DAILY
UA92                 SFO  BOS 08:00 16:27 D8S      B     DAILY
UA194/UA352          SFO  BOS 06:20 16:19 D10/D8S  BL    DAILY
UA982/UA352          OAK  BOS 06:25 16:19 733/D8S  BL    DAILY
UA354                OAK  BOS 11:05 20:56 72S      LD    DAILY
UA820/UA354          SFO  BOS 11:08 20:56 D10/72S  LD    DAILY
```

[End Result: 005]

[UtteranceID:] 006
[Timestamp: Sent speech for utterance 006 at 16:11:34]

[Timestamp: BBN sent transcription for utterance 006 at 16:11:59]
[Begin Utterance: 006]
WHAT IS THE ROUND-TRIP FIRST CLASS FARE ON UNITED FROM BOSTON TO SAN FRANCISCO
[End Utterance: 006]

[Timestamp: Sent answer for utterance 006 at 16:12:49]
[Begin Result: 006]

```
FARE ID  ONE-WAY   ROUND     FARE  RESTRICTION
         COST      TRIP      CODE
-------- --------- --------- ----- -----------
7100570    $836.00 $1,672.00 F     --
7100132    $860.00 $1,720.00 F     --
```

[End Result: 006]

[UtteranceID:] 007
[Timestamp: Sent speech for utterance 007 at 16:13:42]
[Timestamp: BBN sent transcription for utterance 007 at 16:14:01]
[Begin Utterance: 007]
WHAT IS THE ROUND-TRIP THRIFT FARE ON U S AIR FROM BOSTON TO SAN FRANCISCO
[End Utterance: 007]

[Timestamp: Sent answer for utterance 007 at 16:16:24]
[Begin Result: 007]

```
FARE ID  ONE-WAY   ROUND     FARE  RESTRICTION
         COST      TRIP      CODE
-------- --------- --------- ----- -----------
7100137    $573.00 $1,146.00 Y     --
```

[End Result: 007]

[UtteranceID:] 008
[Timestamp: Sent speech for utterance 008 at 16:17:00]
[Timestamp: BBN sent transcription for utterance 008 at 16:17:13]
[Begin Utterance: 008]
ON UNITED

[End Utterance: 008]

[Timestamp: Sent answer for utterance 008 at 16:18:27]
[Begin Result: 008]

```
FARE ID  ONE-WAY   ROUND     FARE  RESTRICTION
         COST      TRIP      CODE
-------- --------- --------- ----- -----------
7100142    $189.00   $378.00 QX    AP/80
7100567    $189.00   $378.00 QX    AP/80
```

[End Result: 008]

[UtteranceID:] 009
[Timestamp: Sent speech for utterance 009 at 16:19:08]
[Timestamp: BBN sent transcription for utterance 009 at 16:19:22]
[Begin Utterance: 009]
ON DELTA
[End Utterance: 009]

[Timestamp: Sent answer for utterance 009 at 16:19:47]
[Begin Result: 009]

```
FARE ID  ONE-WAY   ROUND     FARE  RESTRICTION
         COST      TRIP      CODE
-------- --------- --------- ----- -----------
7100137    $573.00 $1,146.00 Y     --
```

[End Result: 009]

[UtteranceID:] 00A
[Timestamp: Sent speech for utterance 00A at 16:20:04]
[Timestamp: BBN sent transcription for utterance 00A at 16:20:11]
[Begin Utterance: 00A]
ON TWA
[End Utterance: 00A]

[Timestamp: Sent answer for utterance 00A at 16:20:42]
[Begin Result: 00A]

```
FARE ID  ONE-WAY   ROUND     FARE  RESTRICTION
         COST      TRIP      CODE
-------- --------- --------- ----- -----------
7100142    $189.00   $378.00 QX    AP/80
```

[End Result: 00A]

[UtteranceID:] 00B
[Timestamp: Sent speech for utterance 00B at 16:21:00]
[Timestamp: BBN sent transcription for utterance 00B at 16:21:09]
[Begin Utterance: 00B]
ON AMERICAN AIRLINES
[End Utterance: 00B]

[Timestamp: Sent answer for utterance 00B at 16:21:48]
[Begin Result: 00B]

```
FARE ID  ONE-WAY   ROUND     FARE  RESTRICTION
         COST      TRIP      CODE
-------- --------- --------- ----- -----------
7100138    $508.00 $1,016.00 B     --
```

[End Result: 00B]

[UtteranceID:] 00C
[Timestamp: Sent speech for utterance 00C at 16:22:03]
[Timestamp: BBN sent transcription for utterance 00C at 16:22:17]
[Begin Utterance: 00C]
EXPLAIN RESTRICTION AP80
[End Utterance: 00C]

[Timestamp: Sent answer for utterance 00C at 16:22:35]
[Begin Result: 00C]

```
RESTRICTION APPLICATION
----------- ------------------------------------------
AP/80       FARES APPLY TO/FROM INTERMEDIATE POINTS
```

[End Result: 00C]

[UtteranceID:] 00D
[Timestamp: Sent speech for utterance 00D at 16:23:26]
[Timestamp: BBN sent transcription for utterance 00D at 16:23:43]
[Begin Utterance: 00D]
LIST ALL UNITED FLIGHTS FROM BOSTON TO SAN FRANCISCO
[End Utterance: 00D]

[Timestamp: Sent answer for utterance 00D at 16:23:50]
[Begin Result: 00D]

```
AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT
                              TIME  TIME  CODES          DAYS
-------------------- ---- --- ----- ----- -------- ----- ---------
UA21                 BOS  SFO 08:00 11:31 D10      B     DAILY
UA343/UA551          BOS  OAK 08:40 13:39 D8S/727  BL    SU
UA93                 BOS  SFO 17:45 21:27 D8S      D     DAILY
UA201/UA551          BOS  OAK 08:38 13:39 D8S/727  BL    NOT SU
UA281/UA297          BOS  SFO 17:20 22:14 D8S/72S  D     DAILY
UA343                BOS  SFO 08:40 13:36 D8S      BL    SU
UA281/UA673          BOS  OAK 17:20 22:05 D8S/733  D     DAILY
UA201/UA343          BOS  SFO 08:38 13:36 D8S/D8S  BL    NOT SU
```

[End Result: 00D]

[UtteranceID:] 00E
[Timestamp: Sent speech for utterance 00E at 16:25:01]
[Timestamp: BBN sent transcription for utterance 00E at 16:25:21]
[Begin Utterance: 00E]
EXPLAIN FARE ID
[End Utterance: 00E]

[Timestamp: BBN Sent query failure for utterance 00E at 16:26:41]
[Begin error: 00E]
Sorry, the system was unable to answer your question.
[End Error: 00E]

[UtteranceID:] 00F
[Timestamp: Sent speech for utterance 00F at 16:27:29]
[Timestamp: BBN sent transcription for utterance 00F at 16:27:49]
[Begin Utterance: 00F]
EXPLAIN A/C CODES
[End Utterance: 00F]

[Timestamp: Sent answer for utterance 00F at 16:28:20]
[Begin Result: 00F]

```
PROPULSION AIRCRAFT AIRCRAFT DESCRIPTION
---------- -------- --------------------
TURBOPROP  DH8      BOEING CANADA DHC...
JET        727      BOEING 727 PASSEN...
JET        72S      BOEING 727-200 (A...
JET        73S      BOEING 737 PASSEN...
JET        733      BOEING 737-300
JET        734      BOEING 737-400
JET        74M      BOEING 747 MIXED ...
JET        757      BOEING 757-200 PA...
JET        767      BOEING 767 (ALL S...
JET        763      BOEING 767-300/300ER
TURBOPROP  J31      BRITISH AEROSPACE...
TURBOPROP  SWM      FAIRCHILD (SWEARI...
JET        F28      FOKKER F28 FELLOW...
JET        L10      LOCKHEED L1011 (A...
JET        L15      LOCKHEED L1011-50...
JET        D8S      MCDONNELL DOUGLAS...
JET        D9S      MCDONNELL DOUGLAS...
JET        D10      MCDONNELL DOUGLAS...
JET        M80      MCDONNELL DOUGLAS...
TURBOPROP  SH3      SHORTS 330 PASSENGER
TURBOPROP  SH6      SHORTS 360
JET        100      FOKKER 100
```

[End Result: 00F]

[UtteranceID:] 00G
[Timestamp: Sent speech for utterance 00G at 16:29:03]
[Timestamp: BBN sent transcription for utterance 00G at 16:29:35]
[Begin Utterance: 00G]
LIST ALL UNITED FLIGHTS FROM BOSTON TO SAN FRANCISCO
WITH FARE CODE QX
[End Utterance: 00G]

[Timestamp: Sent answer for utterance 00G at 16:29:58]
[Begin Result: 00G]

```
AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT
                              TIME  TIME  CODES          DAYS
-------------------- ---- --- ----- ----- -------- ----- ---------
UA21                 BOS  SFO 08:00 11:31 D10      B     DAILY
UA93                 BOS  SFO 17:45 21:27 D8S      D     DAILY
UA201/UA551          BOS  OAK 08:38 13:39 D8S/727  BL    NOT SU
UA281/UA297          BOS  SFO 17:20 22:14 D8S/72S  D     DAILY
UA281/UA673          BOS  OAK 17:20 22:05 D8S/733  D     DAILY
UA201/UA343          BOS  SFO 08:38 13:36 D8S/D8S  BL    NOT SU
```

[End Result: 00G]

[UtteranceID:] 00H
[Timestamp: Sent speech for utterance 00H at 16:31:11]
[Timestamp: BBN sent transcription for utterance 00H at 16:31:48]
[Begin Utterance: 00H]
LIST ALL FLIGHTS ON UNITED FROM SAN FRANCISCO TO BOSTON WITH FARE CODE QX
[End Utterance: 00H]

[Timestamp: Sent answer for utterance 00H at 16:32:00]
[Begin Result: 00H]

```
AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT
                              TIME  TIME  CODES          DAYS
-------------------- ---- --- ----- ----- -------- ----- ---------
UA20                 SFO  BOS 13:40 22:02 D10      L     DAILY
UA92                 SFO  BOS 08:00 16:27 D8S      B     DAILY
UA194/UA352          SFO  BOS 06:20 16:19 D10/D8S  BL    DAILY
UA982/UA352          OAK  BOS 06:25 16:19 733/D8S  BL    DAILY
UA354                OAK  BOS 11:05 20:56 72S      LD    DAILY
```

[End Result: 00H]

[UtteranceID:] 00I
[Timestamp: Sent speech for utterance 00I at 16:33:14]
[Timestamp: BBN sent transcription for utterance 00I at 16:33:29]
[Begin Utterance: 00I]
LIST THE NUMBER OF STOPS FOR EACH OF THOSE FLIGHTS
[End Utterance: 00I]

[Timestamp: Sent answer for utterance 00I at 16:33:45]

[Begin Result: 00I]

```
STOPS AIRLINE + FLIGHT #   FROM TO  DEPT  ARRV  A/C      MEALS FLIGHT
                                    TIME  TIME  CODES          DAYS
----- -------------------- ---- --- ----- ----- -------- ----- ---------
1     UA194/UA352          SFO  BOS 06:20 16:19 D10/D8S  BL    DAILY
0     UA92                 SFO  BOS 08:00 16:27 D8S      B     DAILY
1     UA982/UA352          OAK  BOS 06:25 16:19 733/D8S  BL    DAILY
1     UA354                OAK  BOS 11:05 20:56 72S      LD    DAILY
0     UA20                 SFO  BOS 13:40 22:02 D10      L     DAILY
```

[End Result: 00I]

[Timestamp: BBN End log at 16:35:48]

11.5.2 The Utterances

Here are the utterances from the preceding sample session, in lexicalized SNOR format. Each utterance is followed by its utterance ID in the ATIS2 corpus.

WHICH AIRLINES FLY FROM BOSTON TO SAN FRANCISCO (e30011sx)
WHICH OF THOSE FLIGHTS HAVE FIRST CLASS SERVICE (e30021sx)
LIST THOSE FLIGHTS LEAVING BEFORE NINE A M DAILY (e30031sx)
WHICH OF THOSE FLIGHTS ARE NONSTOP (e30041sx)
LIST ALL FLIGHTS ON UNITED FROM SAN FRANCISCO TO BOSTON (e30051sx)
WHAT IS THE ROUND TRIP FIRST CLASS FARE ON UNITED FROM BOSTON
TO SAN FRANCISCO (e30061sx)
WHAT IS THE ROUND TRIP THRIFT FARE ON U S AIR FROM BOSTON
TO SAN FRANCISCO (e30071sx)
ON UNITED (e30081sx)
ON DELTA (e30091sx)
ON T W A (e300a1sx)
ON AMERICAN AIRLINES (e300b1sx)
EXPLAIN RESTRICTION A P EIGHTY (e300c1sx)
LIST ALL UNITED FLIGHTS FROM BOSTON TO SAN FRANCISCO (e300d1sx)
EXPLAIN FARE I D (e300e1sx)
EXPLAIN A C CODES (e300f1sx)

LIST ALL UNITED FLIGHTS FROM BOSTON TO SAN FRANCISCO
    WITH FARE CODE Q X (e300g1sx)
LIST ALL FLIGHTS ON UNITED FROM SAN FRANCISCO TO BOSTON
    WITH FARE CODE Q X (e300h1sx)
LIST THE NUMBER OF STOPS FOR EACH OF THOSE FLIGHTS (e300i1sx)

11.6 Final Summary of The HARC System

We conclude this chapter with the descriptions of DELPHI and BYBLOS and of the overall HARC system that were submitted to NIST as part of our participation in the February, 1992 Cross-Site Evaluations of speech recognition, natural language, and spoken language systems. We hope this will give a good overview of the state of our systems at the end of this project.

11.6.1 DELPHI

1) OVERALL SYSTEM DESCRIPTION: The BBN DELPHI natural language processing system uses a grammar based on the definite clause grammar formalism, extended with type declarations, which restrict the range of the possible values of argument, and a limited form of disjunction. It is parsed using a variant of the Graham-Harrison-Russo algorithm, which has been modified to use a trainable agenda to produce a best first parse, rather than all possible parses. While "unification grammars" normally build up semantic representations in tandem with syntactic structure, our system performs only semantic type consistency checking on the fly, and attempts to form WFFs only when all the necessary semantic elements have been found (typically, at the clausal level). A fragment processor produces a semantic representation in those cases in which the parser cannot return a parse. The discourse component of DELPHI uses an extension of Bonnie Webber's discourse entities model to determine antecedents of pronouns and other anaphoric expressions. A domain-specific plan tracker is used to keep track of the state of the discourse and to record contextual information that may be in effect for more than one query (e.g. in the case of ATIS: origin, destination, flight day, etc.) The semantic representation is in a version of MRL, a Meaning Representation Language first used in LUNAR. This is translated into ERL for database retrievals.

2) TRAINING DATA: For the February 1992 evaluation test, the main training data used were the NL utterances and annotations in the third through eighth shipments of annotated data for atis2 (answer.910830 through answer.911030; 1933 annotated utterances and 2555 total utterances). At first, we used the ninth through thirteenth shipments of annotated

data (answer.911108 through answer.911217) as development test. Eventually, we added all but answer.911217 to a second training set (1405 annotated utterances and 1746 total utterances), retaining answer.911217 (365 annotated utterances and 455 total utterances) and adding the October 1991 dry run material (288 annotated utterances and 361 total utterances) as development test. Finally, these two small corpora were added to the training as well. The grammar and lexicon had previously been trained on the class A sentences from the atis0 data release, the class A and D1 utterances from the atis2 release collected by TI, and the first two shipments of annotated cross-site data for atis2. The later atis2 data releases were used as incremental additions to this.

3) LEXICON DESCRIPTION: The DELPHI system for ATIS uses multiple lexicons: a domain independent core lexicon of a little over 300 entries (loaded in for any domain); a domain independent lexicon of approximately 50 common terms for database lookup applications (loaded in for any database application); and a domain specific lexicon of nearly 2000 lexical entries for ATIS. The DELPHI lexicons contain collocations as independent lexical entries; these are processed by DELPHI as if they were entered directly as rules in the grammar. Inflected forms do not have separate entries, but are either derived by morphology in the case of regular lexical items, or are entered as part of the lexical entry for the base form, for irregularly inflected lexical items.

4) NEW CONDITIONS FOR THIS EVALUATION: The DELPHI system for the February 1992 evaluation was different from the system used for the October 1991 dry run in several ways. The use of labelled arguments in grammar rules, previously reported in our work on mapping unit subcategorization, was extended to the grammar as a whole. The fragment combiner was extended, and many changes were made to the discourse component and task tracker.

5) REFERENCES:

Ayuso, D. "Discourse Entities in Janus", *Proceedings of ACL 27*, Association for Computational Linguistics, Murray Hill, NJ, pp. 243–250.

Bobrow, R. "Statistical Agenda Parsing" *Proceedings Speech and Natural Language Workshop February 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991, pp. 222–224.

Bobrow, R., Ingria, R. and Stallard, D. "Syntactic and Semantic Knowledge in the DELPHI Unification Grammar". *Proceedings Speech and Natural Language Workshop June 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990, pp. 230–236.

Bobrow, R., Ingria, R. and Stallard, D. "The Mapping Unit Approach to Subcategorization". *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, Califor-*

nia, February 19–22, 1991, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991, pp. 185–189.

11.6.2 BYBLOS

1) OVERALL SYSTEM DESCRIPTION:
The Feb 92 BBN BYBLOS system used 45 spectral features to model speech: 14 cepstra and their 1st and 2nd differences, plus normalized energy and its 1st and 2nd difference. The HMM observation densities were modeled as tied Gaussian mixtures defined by K-means clustering and fixed during training. Context-dependent phonetic HMMs (including those occurring across word boundaries) were constructed from triphones, left-diphones, right-diphones, and context-independent phonemes that were trained jointly in forward-backward. Phoneme-dependent cooccurrence smoothing matrices were estimated from the trained triphone context models and applied to all context-dependent observation densities.

The BYBLOS recognizer used a time-synchronous, 3-pass algorithm that utilizes progressively more detailed models in an efficient manner [1]. The 1st pass proceeds forward (in time) through the data saving ending times and scores for the top-scoring partial word-sequences. The 2nd pass proceeds backward computing the top N best word-sequences, taking advantage of the word-sequence-ending scores from the first pass to prune aggressively based on the score for the entire hypothesis under consideration. In the 3rd (forward) pass, the N-best hypotheses are reordered using the system's most detailed models, including cross-word-boundary triphone HMMs and higher order ngram statistical language models. The top scoring answer after the rescoring pass constitutes the speech recognition result.

2) ACOUSTIC TRAINING:
8000 spontaneously produced utterances were used for acoustic training. All were from ATIS2 MADCOW data. The training was partitioned in order to train gender-dependent models. 800 additional utterances were held-out for development testing.

3) GRAMMAR TRAINING:
A bigram class grammar was used for the Nbest. The rescoring pass used a trigram language model. The grammars included 1050 word-classes and were smoothed with backoff probabilities. They were trained on the 14500 ATIS sentences in the ATIS0, ATIS1, and ATIS2 subcorpora. The bigram perplexity was 14 on training and 18 on the development test, while the trigram perplexity was 7 on the training, 13 on the development test.

4) RECOGNITION LEXICON DESCRIPTION:
The recognition dictionary had 1881 words. Only 7 words had multiple pronunciations. 15 different nonspeech events were treated as words in the dictionary.

5) NEW CONDITIONS FOR THIS EVALUATION:
This is the first time that a large quantity of spontaneous ATIS speech has been available to train the HMM and language model parameters. We have used the trigram grammar effectively on this task for the first time.

6) REFERENCES:
[1] "New Uses for N-Best Sentence Hypotheses Within the BYBLOS Speech Recognition System", ICASSP-92

11.6.3 HARC

(1) OVERALL SYSTEM DESCRIPTION: The SLS system used for the February 1992 evaluation used the BYBLOS system for speech recognition and the DELPHI system for natural language processing (see individual descriptions of these systems for full details). DELPHI processed the N-best lists produced by BYBLOS in the following manner: a parameter S, indicating the maximum depth of search into the N-best list that DELPHI would perform, was set. These S utterances were first processed by DELPHI with fragment processing disabled until a database retrieval was made. If no database retrieval was made, DELPHI iterated over the same S utterance, with fragment processing enabled. For this evaluation, S was set at 5.

(2) TRAINING DATA: See individual descriptions of BYBLOS and DELPHI for the training sets used for SPREC and NL separately. We developed our interface strategy on the basis of combined SLS data from the October '91 dry run and a subset of the NL training data (approximately 700 sentences). We used a development test set of approximately 350 utterances comprising the intersection of the development test sets for SPREC and NL.

(3) NEW CONDITIONS FOR THIS EVALUATION: See individual descriptions of BYBLOS and DELPHI.

(4) REFERENCES: See individual descriptions of BYBLOS and DELPHI.

Chapter 12

Publications and Presentations

12.1 Publications

Asadi, Ayman, Richard Schwartz, and John Makhoul (1989) "Automatic Detection of New Words in a Large Vocabulary Continuous Speech Recognition System", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 263–265.

Asadi, A., R. Schwartz, and J. Makhoul (1990) "Automatic Detection of New Words in a Large Vocabulary Continuous Speech Recognition System", IEEE International Conference on Acoustics, Speech, and Signal Processing in Albuquerque, NM, April 3–6, 1990.

Asadi, A., R. Schwartz, and J. Makhoul (1991) "Automatic modeling for adding new words to a large-vocabulary continuous speech recognition system", IEEE International Conference on Acoustics, Speech, and Signal Processing in Toronto, Canada, May 14–17, 1991.

Austin, S., D. Ayuso, M. Bates, R. Bobrow, R. Ingria, J. Makhoul, P. Placeway, R. Schwartz, and D. Stallard (1991) "BBN HARC and DELPHI Results on the ATIS Benchmarks—February 1991", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 112–115.

Austin, Steve, Chris Barry, Yen-Lu Chow, Alan Derr, Owen Kimball, Francis Kubala, John Makhoul, Paul Placeway, William Russell, Richard Schwartz, and George Yu (1989) "Improved HMM Models for High Performance Speech Recognition", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts,* *October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 249–255.

Austin, Steve, Pat Peterson, Paul Placeway, Richard Schwartz, and Jeff Vandergrift (1990) "Toward a Real-Time Spoken Language System Using Commercial Hardware", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 72–77.

Austin, S., R. Schwartz, and P. Placeway (1991) "The forward-backward search algorithm", IEEE International Conference on Acoustics, Speech, and Signal Processing in Toronto, Canada, May 14–17, 1991.

Bates, Madeleine and Damaris Ayuso (1991) "A Proposal for Incremental Dialogue Evaluation", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 319–322.

Bates, Madeleine and Sean Boisen (1991) "A Developing Methodology for the Evaluation of Spoken Language Systems", in Jeannette G. Neal and Sharon M. Walter, eds., *Natural Language Processing Systems Evaluation Workshop*, University of California, Berkeley, CA, 18 June 1991, pp. 19–29.

Bates, M., R. Bobrow, S. Boisen, R. Ingria, and D. Stallard (1990) "BBN ATIS System Progress Report—June 1990", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 125–126.

Bates, Madeleine, Sean Boisen, and John Makhoul (1990) "Developing an Evaluation Methodology for Spoken Language Systems", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 102–108.

Bates, Madeleine, Dan Ellard, Pat Peterson, and Varda Shaked (1991) "Using Spoken Language to Facilitate Transportation Planning", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 217–220.

Black, E., S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R, Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski (1991) "A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 306–311.

Bobrow, Robert (1991) "Statistical Agenda Parsing", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 222–224.

Bobrow, R. and Lance Ramshaw (1990) "On Deftly Introducing Procedural Elements into Unification Parsing", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 237–240.

Bobrow, R., Robert Ingria, and David Stallard (1990) "Syntactic and Semantic Knowledge in the DELPHI Unification Grammar", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 230–236.

Bobrow, Robert, Robert Ingria, and David Stallard (1991) "The Mapping Unit Approach to Subcategorization", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 185–189.

Boisen, S. (1989) *The SLS Personnel Database (Release 1)*, SLS Notes No. 2, BBN Systems and Technologies Corporation, 1989.

Boisen, Sean, Lance Ramshaw, Damaris Ayuso, and Madeleine Bates (1989) "A Proposal for SLS Evaluation", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 135–146

Chow, Y-L. (1990) "Maximum Mutual Information Estimation of HMM Parameters for Continuous Speech Recognition Using the N-Best Algorithms", IEEE International Conference on Acoustics, Speech, and Signal Processing in Albuquerque, NM, April 3–6, 1990.

Haas, Andrew (1989) "A Generalization of the Offline-Parsable Grammars", in *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ. pp.

Harrison, Philip, Steven Abney, Ezra Black, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Donald Hindle, Robert Ingria, Mitch Marcus, Beatrice Santorini, Tomek Strzalkowski (1991) "Evaluating Syntax Performance of Parser/Grammars of English", in Jeannette G. Neal and Sharon M. Walter, eds., *Natural Language Processing Systems Evaluation Workshop*, University of California, Berkeley, CA, 18 June 1991, pp. 71–77.

Ingria, Robert J. P. (1989) "Simulation of Language Understanding: Lexical Recognition", in *Computational Linguistics: An International Handbook on Computer Oriented Language Research and Applications*, Walter de Gruyter · Berlin · New York, pp. 336–347.

Ingria, Robert J. P. (1990) "The Limits of Unification", in *28th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ. pp. 194–204.

Ingria, Robert and Lance Ramshaw (1989) "Porting to New Domains Using the Learner", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 241–244.

Ingria, Robert and David Stallard (1989) "A Computational Mechanism for Pronominal Reference", in *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ. pp. 262–271.

Kubala, F., S. Austin, C. Barry, J. Makhoul, P. Placeway, R. Schwartz (1991) "BYBLOS Speech Recognition Benchmark Results", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 77–82.

Kubala, F. and R. Schwartz (1991) "A new paradigm for speaker-independent training", IEEE International Conference on Acoustics, Speech, and Signal Processing in Toronto, Canada, May 14–17, 1991.

Ramshaw, L. (1989) *Manual for SLS ERL (Release 1)*, SLS Notes No. 3, BBN Systems and Technologies Corporation, 1989.

Schwartz, Richard and Steve Austin (1990) "Efficient, High-Performance Algorithms for N-Best Search", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 6–11.

Schwartz, R. and S. Austin (1991) "A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses", IEEE International Conference on Acoustics, Speech, and Signal Processing in Toronto, Canada, May 14–17, 1991.

Schwartz, R., and Y-L. Chow (1990) "The N-Best Algorithm: An Efficient and Exact Procedure for Finding the N Most Likely Sentence Hypotheses", IEEE International Conference on Acoustics, Speech, and Signal Processing in Albuquerque, NM, April 3–6, 1990.

Stallard, David "Unification-Based Semantic Interpretation in the BBN Spoken Language System", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 39–46.

12.2 Abstracts Accepted

Bobrow, R., Robert Ingria, and David Stallard "Syntactic/Semantic Coupling in the DELPHI System", Fifth DARPA Speech and Natural Language Workshop, Arden House, February 23–26, 1992.

Bobrow, R., and David Stallard "Fragment Processing in the DELPHI System", Fifth DARPA Speech and Natural Language Workshop, Arden House, February 23–26, 1992.

12.3 Presentations

Bates, Madeleine (1991) talk presented at the DARPA SLS Mid-term Meeting, Carnegie Mellon University, October 21, 1991.

Ingria, Robert J. P. (1989) "Grammar Evaluation in the BBN Spoken Language System", ALLC/ICCH conference on "The Dynamic Text", Toronto, June 6, 1989.

Ingria, Robert J. P. (1990a) "Grammar Development and Evaluation in the BBN Spoken Language System", talk presented at the University of Chicago Center for Information and Language Studies, Chicago, May 21, 1990.

Ingria, Robert J. P. (1990b) "Grammar Engineering in DELPHI", talk presented at the Grammar Engineering Workshop, University of Saarbrücken, June 22, 1990.

Ingria, Robert J. P. (1991a) "Experiments with Unification Grammar", talk presented at Cambridge University, October 3, 1991 and the University of Tübingen, October 11, 1991.

Ingria, Robert J. P. (1991b) "BBN ATIS Data Collection", talk presented at the DARPA SLS Mid-term Meeting, Carnegie Mellon University, October 21, 1991.

Kubala, Francis (1991) talk presented at the DARPA SLS Mid-term Meeting, Carnegie Mellon University, October 21, 1991.

12.4 Meeting and Committee Participation

We participated in (* chaired) numerous committees, including MADCOW, Principles of Interpretation, *Common Lexicon, *Data Formats, Semantic Evaluation, and Workshop Planning. We attended the following meetings of SLS and other community related committees:

- On April 27–28, 1989, we attended a meeting at NIST on a new common task domain and on Spoken Language evaluation criteria.

- On August 23, 1989, Bob Ingria attended a meeting at Bell Labs of the Steering Committee of the TreeBank project, whose purpose is to collect a large annotated corpus of text for the community. Issues discussed include how the corpus should be tagged, and how those tags should be entered while ensuring some degree of consistency.
- On November 15–16, 1989, we attended the DARPA planning meeting in Washington. The main topic of discussion was the new common task domain and performance testing procedures.
- On February 28 and March 1, 1990, John Makhoul and Lyn Bates attended the DARPA Coordinating Committee meeting at TI in Dallas, Texas. The main subject was data collection and performance evaluation for the new ATIS domain.
- On July 26–27, 1990, we attended the SLS Coordinating Committee meeting held at Dragon Systems in Newton, MA. We gave a presentation on our progress and participated in the discussions.
- On August 2, 1990, Bob Ingria attended a meeting at Bell Labs of the Steering Committee of the TreeBank project. The topic of the meeting was the best methodology for adding syntactic bracketing to the tagged corpora, so as to provide useful syntactic information while maintaining consistency and a high level of throughput.
- On November 2–3, 1990, Lyn Bates attended and participated in the DARPA SLS Coordinating Committee meeting held in Palo Alto.
- On March 18, 1991, we attended the Corpus Production and Evaluation Committee (CPAC) held at MIT.
- On March 19–20, 1991, we attended the Coordinating Committee Meeting held at MIT.
- On July 29–30 1991, we participated in the DARPA SLS Coordinating Committee Meeting, held at AT&T Bell Laboratories in Murray Hill, NJ.

We hosted visits from government personnel and other members of the DARPA Speech and Natural Language community on the following occasions:

- On April 14, 1989, Charles Wayne (DARPA) and David Pallett (NIST) visited BBN for presentations about our recent Spoken Language Systems work. They were also given demonstrations of and participated in data acquisition by our Wizard of Oz simulations for the personnel database domain.
- On November 9, 1989, we hosted Cliff Weinstein, Victor, Zue, and Jim Glass, to demonstrate the personnel database and our Wizard of Oz system.

- On March 6, 1990, Charles Wayne visited BBN in which we reviewed progress on our spoken language system work.
- On August 1, 1990, Charles Wayne of DARPA visited BBN to review our spoken language systems work. We demonstrated, for the first time, a speaker-independent version of BYBLOS running in near real-time on a Sun 4 using a fully connected statistical grammar. We also made several technical presentations and discussed contract matters.
- On November 7, 1990, we gave a demonstration of our real–time HARC spoken language system running in the DART domain to Tom Crystal of DARPA/ISTO.

We made the following site visits:

- On August 3, 1989, we visited the Spoken Language Systems group at MIT, and saw demonstrations of their Voyager application. We discussed the differences between our data collection paradigms, where our Wizard of Oz approach obtains data to be used for research purposes over a long period of time, while their frequent data collections by students are intended for short-term use.
- On September 6–8, 1989, we visited Texas Instruments to deliver our Wizard of Oz system to be used for data collection, and trained them in its use.

We participated in the following DARPA Speech and Natural Language Workshops:

- On October 15–18, 1989, we attended the Second DARPA Speech and Natural Language Workshop held at Cape Cod, and presented papers on unification-based semantic interpretation, evaluation of database query systems, developing statistical class grammars from limited data, the optimal N-best algorithm, and automatic detection of new words.
- On June 24–27, 1990, we attended the Third DARPA Speech and Natural Language Workshop held in Hidden Valley, PA, and presented papers on SLS evaluation, cor bining syntax and semantics in a unification grammar, on efficient unification grammar parsing, the N-Best algorithm, and the development of a real-time system.
- On February 19–22, 1991, we attended the Fourth DARPA Speech and Natural Language Workshop held at Asilomar and presented papers on our mapping units approach to subcategorization, our best first statistical parsing algorithm, a proposal for incremental dialogue evaluation, and a discussion of our results on the speech, natural language, and spoken language benchmark evaluation.

As part of our participation in the technical life of the natural language, speech recognition, and spoken language systems communities, we attended the following conferences:

- On June 5–8, 1989, we attended the MUCK-II workshop on Message Understanding at NOSC as observers. The workshop was useful in providing information about that the state of the art is in message processing.
- On June 26–29, 1989, we attended the 27th Annual Meeting of the Association for Computational Linguistics at the University of British Columbia, and presented papers on our work in the areas of syntax, semantics, and discourse processing.
- On September 26–27, 1989, we attended the RADC Natural Language Workshop, where we made presentation on knowledge acquisition and Spoken Language System evaluation.
- On November 29–December 1, 1989, we attended the Natural Language Symposium hosted by BBN, where we made a presentation on alternatives in Spoken Language Systems research.
- On June 6–9, 1990, we attended the 28th Annual Meeting of the Association for Computational Linguistics held in Pittsburgh, and presented a paper on the limits of unification.
- On April 3–6, 1990, we attended the IEEE International Conference on Acoustics, Speech, and Signal Processing in Albuquerque, NM and presented papers on the N-Best algorithm and detection of out-of-vocabulary words.
- On May 14–17, 1991, we attended the IEEE International Conference on Acoustics, Speech, and Signal Processing in Toronto, Canada and presented the papers on the N-best algorithm, the forward-backward search algorithm, adding new words to a large vocabulary continuous speech recognition system, and a new paradigm for speaker independent training.
- On June 19–21, 1991, we attended the 29th Annual Meeting of the Association for Computational Linguistics in Berkeley, California. At a pre-conference Workshop on Evaluation of Natural Language Processing Systems, we presented a paper on a developing methodology for the evaluation of spoken language systems.
- On October 21–22, 1991, Bob Ingria participated in the workshop on Open Lexical and Text Resources held at the University of Pennsylvania.
- On November 28–29, 1991, Bob Ingria participated in the Syntax Evaluation Workshop held at the University of Pennsylvania. This workshop produced the first workable proposal for evaluating the syntactic output of natural language parsing systems.

Bibliography

[1] Abney, Steven P. (1990) "Rapid Incremental Parsing with Repair", in *Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research: Electronic Text Research, Proceedings of the Conference*, October 28-30, 1990, University of Waterloo, Waterloo, Ontario, Canada, pp. 1–9.

[2] Ades, A. E. and Mark J. Steedman (1982) "On the Order of Words", *Linguistics and Philosophy* 44.3, pp. 517–558.

[3] Aubert, X.L. (1990) "A Fast Lexical Selection Strategy for Large Vocabulary Continuous Speech Recognition", *NATO Advanced Study Institute on Speech Recognition and Understanding* Cetraro, Italy, July 1990.

[4] Austin, S., P. Peterson, P. Placeway, R. Schwartz, J. Vandergrift (1990) "Toward a Real-Time Spoken Language System Using Commercial Hardware", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 72–77.

[5] Austin, S., Schwartz, R., and Placeway, P. (1991) "The Forward-Backward Search Strategy for Real-Time Speech Recognition", *Proceedings of the ICASSP 91*, Toronto, Canada, May 1991, pp. 697-700

[6] Austin, S., Makhoul, J., Schwartz, R., Zavaliagkos, G. (1991) "Continuous Speech Recognition Using Segmental Neural Nets", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 249–253.

[7] Austin, S., Zavaliagkos, G., Makhoul, J., and R. Schwartz (1992) "Speech Recognition using Segmental Neural Nets", *Proceedings of the ICASSP 92*.

[8] Ayuso, D. (1989) "Discourse Entities in Janus", in *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ. pp. 243–250.

[9] Ayuso, D., Bobrow, R., MacLaughlin, D., Meteer, M., Ramshaw, L., Schwartz, R., and Weischedel, R. (1990) "Towards Understanding Text with a Very Large Vocabulary", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 354–358.

[10] Bahl, L. R., S. Das, P. V. deSouza, M. Epstein, R. L. Mercer, B. Merialdo, D. Nahamoo, M. A. Picheny, and J. Powell (1990) "Automatic Phonetic Baseform Determination", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 179–184.

[11] Bahl, L.R., de Souza, P., Gopalakrishnan, P.S., Kanevsky, D., and D. Nahamoo (1990) "Constructing Groups of Acoustically Confusable Words", *Proceedings of the ICASSP 90*, April, 1990.

[12] Bates, Madeleine and Damaris Ayuso (1991) "A Proposal for Incremental Dialogue Evaluation", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 319–322.

[13] Bates, M., R. Bobrow, S. Boisen, R. Ingria, and D. Stallard (1990) "BBN ATIS System Progress Report—June 1990", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 125–126.

[14] Bates, Madeleine and Sean Boisen (1991) "A Developing Methodology for the Evaluation of Spoken Language Systems", in Jeannette G. Neal and Sharon M. Walter, eds., *Natural Language Processing Systems Evaluation Workshop*, University of California, Berkeley, CA, 18 June 1991, pp. 19–29.

[15] Bates, Madeleine, Sean Boisen, and John Makhoul (1990) "Developing an Evaluation Methodology for Spoken Language Systems", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 102–108.

[16] Bellegarda, J. and D. Nahamoo (1989) "Tied mixture continuous parameter models for large vocabulary isolated speech recognition" *Proceedings of the ICASSP 89.*

[17] Bisiani, R. (1989) "Plans for PLUS hardware", presentation at the DARPA Speech and Natural Language Workshop, Cape Cod, Massachusetts, October 15–18, 1989.

[18] Black, E., S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R, Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski (1991) "A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars", in *Speech and Natural Language: Proceedings*

of a Workshop Held at Pacific Grove, California, February 19–22, 1991, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 306–311.

[19] Bobrow, Robert (1991) "Statistical Agenda Parsing", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 222–224.

[20] Bobrow, R., Robert Ingria, and David Stallard (1990) "Syntactic and Semantic Knowledge in the DELPHI Unification Grammar", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 230–236.

[21] Bobrow, Robert, Robert Ingria, and David Stallard (1991) "The Mapping Unit Approach to Subcategorization", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 185–189.

[22] Bobrow, R.J., Ingria, R. and Stallard, D. (1992) *Syntactic/Semantic Coupling in the DELPHI System* to appear in the Proceedings of the Speech and Natural Language Workshop, February, 1992.

[23] Bobrow, R. and Lance Ramshaw (1990) "On Deftly Introducing Procedural Elements into Unification Parsing", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 237–240.

[24] Boisen, S., Y. Chow, A. Haas, R. Ingria, S. Roucos, R. Scha, D. Stallard, and M. Vilain (1989) *Integration of Speech and Natural Language: Final Report*, Report No. 6991, BBN Systems and Technologies Corporation, Cambridge, Massachusetts.

[25] Boisen, Sean, Yen-Lu Chow, Andrew Haas, Robert Ingria, Salim Roukos, and David Stallard (1989) "The BBN Spoken Language System", in *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania February 21–22, 1989*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 106–111.

[26] Boisen Sean, Lance Ramshaw, Damaris Ayuso, and Madeleine Bates (1989) "A Proposal for SLS Evaluation", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 135–146

[27] Brennan, Susan E., Friedman, Marilyn W., and Pollard, Carl J. (1987) "A Centering Approach to Pronouns", in *25th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 155–162.

[28] Bresnan, Joan (1982) *The Mental Representation of Grammatical Relations*, The MIT Press, Cambridge, Massachusetts.

[29] Chomsky, Noam (1981) *Lectures on Government and Binding*, Foris Publications, Dordrecht, Holland.

[30] Chow, Y.L. (1990) "Maximum Mutual Information Estimation of HMM Parameters for Continuous Speech Recognition Using the N-Best Algorithm," *Proceedings of the ICASSP 90*, Albuquerque, NM, April, 1990.

[31] Chow, Y.L., Dunham, M.O., Kimball, O.A., Krasner, M.A., Kubala, G.F., Makhoul, J., Price, P.J., Roucos, S., and Schwartz, R.M. (1987) "BYBLOS: The BBN Continuous Speech Recognition System". *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Dallas, TX, April 1987, pp. 89–92, Paper No. 3.7.

[32] Chow, Yen-Lu and Richard Schwartz (1989) "The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 199–202.

[33] Dahl, Deborah A., Martha S. Palmer, and Rebecca J. Passonneau (1987) "Nominalizations in *PUNDIT*". *25th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 131–137.

[34] Davidson, D. (1967) "The Logical Form of Action Sentences" in N. Rescher, ed., *The Logic of Decision and Action*, University of Pittsburgh Press, Pittsburgh, pp. 81–95.

[35] Derr, A. and R. Schwartz (1989) "A Statistical Class Grammar for Measuring Speech Recognition Performance", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 147–149

[36] Emonds, Joseph E. (1976) *A Transformational Approach to English Syntax: Root, Structure-Preserving, and Local Transformations*, Academic Press, New York.

[37] Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag (1985) *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Massachusetts.

[38] Gillick, L. and R. Roth (1990) "A Rapid Match Algorithm for Continuous Speech Recognition", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 170–172.

[39] Graham, Susan L., Michael A. Harrison, and Walter L. Ruzzo (1980) "An Improved Context-free Recognizer", *ACM Transactions on Programming Languages and Systems*, 2.3, pp. 415–461.

[40] Grider, T., H. Mosley, J. Snow, and W. Wilson (1990) "Users Manual for the Dynamic Analytical Replanning Tool (DRAFT)", prepared for BBN by Systems Research and Applications Corporation, 9 November 1990.

[41] Grosz, Barbara J., Joshi, Aravind K., Weinstein, Scott (1983) "Providing a Unified Account of Definite Noun Phrases in Discourse", in *25th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 44–50.

[42] Haas, Andrew "A New Parsing Algorithm for Unification Grammar", *Computational Linguistics*

[43] Halvorsen, P.-K and John Nerbonne (1990) "Unification in the Syntax/Semantics Interface", tutorial presented at the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh, Pennsylvania, 6 June, 1990.

[44] Harrison, Philip, Steven Abney, Ezra Black, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Donald Hindle, Robert Ingria, Mitch Marcus, Beatrice Santorini, Tomek Strzalkowski (1991) "Evaluating Syntax Performance of Parser/Grammars of English", in Jeannette G. Neal and Sharon M. Walter, eds., *Natural Language Processing Systems Evaluation Workshop*, University of California, Berkeley, CA, 18 June 1991, pp. 71–77.

[45] Huang, X.D. and M.A. Jack (1989) "Semi-continuous hidden Markov models for speech recognition" *Computer Speech and Language*. Vol 3, 1989

[46] Ingria, Robert and David Stallard (1989) "A Computational Mechanism for Pronominal Reference", in *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ. pp. 262–271.

[47] Jackson, E., Appelt, D., Bear, J., Moore, R. and Podlozny, A. (1991) "A Template Matcher for Robust NL Interpretation", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 190–194.

[48] Jelinek, F., R. Mercer, and S. Roukos (1990) "Classifying Words for Improved Statistical Language Models", in *Proceedings of the ICASSP 90*, Albuquerque, NM. pp. 621–624.

[49] Johnson, Mark E. (1989) *Attribute-Value Logic and the Theory of Grammar* Center for the Study of Language and Information.

[50] Johnson, R. and M. Rosner (1989) "A rich environment for experimentation with unification grammars". *Fourth Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 182–189.

[51] *Joint Operation Planning System (JOPS) Time Phased Force Deployment Data (TPFDD) and Related Files, Database Specification*, System Planning Manual, SPM D5 143-87, Joint Data Systems Support Center, 1 April 1987.

[52] Karttunen, Lauri (1984) "Features and Values", *Coling84: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 28–33.

[53] Kasami, T. (1965) "An Efficient Recognition and Syntax Analysis Algorithm for Context-Free Languages", Science Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Mass.

[54] Kasper, R. T. (1987) "A Unification Method for Disjunctive Feature Descriptions", in *25th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 235–242.

[55] Kasper, R. T. and Rounds, W. C. (1986) "A Logical Semantics for Feature Structures", in *24th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 257–266.

[56] Katz, Slava M. (1987) "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer", in *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-35, No. 3, March, 1987.

[57] Kubala, F. and R. Schwartz (1991) "A new paradigm for speaker-independent training", IEEE International Conference on Acoustics, Speech, and Signal Processing in Toronto, Canada, May 14–17, 1991.

[58] Lee, C.H. and L.R. Rabiner (1989) "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition," *IEEE Transactions on ASSP*, Vol. 37, No. 11, Nov. 1989, pp. 1649–1658.

[59] Lowerre, B. (1977) *The Harpy Speech Recognition System*, Doctoral Thesis, CMU.

[60] Lucassen, J.M. and R. L. Mercer (1984) "An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms", in *Proceedings of the ICASSP 84*, San Diego, CA.

[61] Marcus, Mitchell P. (1980) *A Theory of Syntactic Recognition for Natural Language*, The MIT Press, Cambridge, Massachusetts.

[62] Mariño, J. and E. Monte (1989) "Generation of Multiple Hypothesis in Connected Phonetic-Unit Recognition by a Modified One-Stage Dynamic Programming Algorithm", *Proc. of the European Conf. on Speech Communication and Technology*, Paris, Sept. 1989, Vol. 2, pp. 408–411.

[63] Mellish, C.S. (1988) "Implementing Systemic Classification by Unification", *Computational Linguistics*, 14.1, pp. 40–51.

[64] "Moens, M., Calder, J., Klein, E., Reape, M., and Zeevat, H. (1989) "Expressing generalizations in unification-based grammar formalisms", in *Fourth Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 174–181.

[65] Montague, Richard (1973) "The Proper Treatment of Quantification in Ordinary English", in J. Hintakka, J. Moravcsik, and P. Suppes, eds., *Approaches to Natural Language': Proceedings of the 1970 Stanford Workship on Grammar and Semantics*, D. Reidel, Dordrecht, pp. 221–242.

[66] Moore, R. C. (1989) "Unification-Based Semantic Interpretation", in *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 33–41.

[67] Moser, Margaret (1983) "An Overview of NIKL", in *Research in Knowlege Representation for Natural Language Understanding, Annual Report: 1 September 1982 to 31 August 1983*, Report No. 5421, BBN, Cambridge, Massachusetts.

[68] Murveit, H. (1989) "Plans for VLSI HMM Accelerator", presentation at the DARPA Speech and Natural Language Workshop, Cape Cod, Massachusetts, October 15–18, 1989.

[69] Murveit, H. (1990) "Integrating Natural Language Constraints into HMM-based Speech Recognition", *Proceedings of the ICASSP 90*, April, 1990.

[70] Ostendorf, M., Kannan, A., Kimball, O., Schwartz, R., Austin, S., Rohlicek, R. (1991) "Integration of Diverse Recognition Methodologies Through Reevaluation of N-Best Sentence Hypotheses", in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19–22, 1991*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 83–87.

[71] Palmer, M. S. (1983) *Driving Semantics for a Limited Domain*. Doctoral Dissertation, University of Edinburgh.

[72] Partee, B. H. (1984) "Compositionality", in *Varieties of Formal Semantics: Proceedings of the fourth Amsterdam Colloquium, September, 1982*, F. Landman and F. Veltman (eds.), FORIS PUBLICATIONS, Dordrecht - Holland/Cinnaminson - U.S.A., 1984, pp. 281–311.

[73] Pereira, Fernando C.N. and Stuart M. Shieber (1987) *Prolog and Natural-Language Analysis*. Center for the Study of Language and Information, Stanford, CA.

[74] Pereira, Fernando C. N. and David H. D. Warren (1980) "Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks", *Artificial Intelligence* 13, pp. 231-278.

[75] Pollack, Martha E. and Fernando C.N. Pereira (1988) "An Integrated Framework for Semantic and Pragmatic Interpretation", in *26th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 75–86.

[76] Price, P., W.M. Fisher, J. Bernstein and D.S. Pallett (1988) "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," *IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, NY, April 1988, pp. 651-654.

[77] Ramshaw, L. (1989) *Manual for SLS ERL (Release 1)*, SLS Notes No. 3, BBN Systems and Technologies Corporation, 1989.

[78] Rohlicek, J.A., Chow, Y-L., and Roucos, S. (1987) "Statistical Language Modeling Using a Small Corpus from an Application Domain", in *Proceedings of the DARPA Speech and Natural Language Workshop* Cambridge, October 1987. Also in *Proceedings of the ICASSP 88*, pp. 267–270, April, 1988.

[79] Ross, John Robert (1967) *Constraints on Variables in Syntax*, Ph.D. dissertation, Massachusetts Institute of Technology.

[80] Ross, John Robert (1967) *Infinite Syntax!*, ABLEX Publishing Corporation, Norwood, New Jersey.

[81] Schabes, Yves, Anne Abeille, and Aravind K. Joshi (1988) "Parsing Strategies with 'Lexicalized' Grammars': Application to Tree Adjoining Grammars", in *COLING Budapest: PROCEEDINGS of the 12th International Conference on Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, pp. 578–583.

[82] Schmolze, James and David Israel (1983) "KL-ONE: Semantics and Classification", in *Research in Knowlege Representation for Natural Language Understanding, Annual Report: 1 September 1982 to 31 August 1983*, Report No. 5421, BBN, Cambridge, Massachusetts.

[83] Schwartz, Richard (1989) "Summary of Session on Hardware for Spoken Language Demonstrations", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 437–438.

[84] Schwartz, R.M., and Austin, S.A. (1990) "Efficient, High-Performance Algorithms for N-Best Search", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 6–11.

[85] Schwartz, R. and S. Austin, (1991) "A Comparison Of Several Approximate Algorithms for Finding Multiple (N-Best) Sentence Hypotheses", *Proceedings of the ICASSP 91*, Toronto, Canada, pp. 701–704.

[86] Schwartz, R.M., Chow, Y., Kimball, O., Roucos, S., Krasner, M., and Makhoul, J. (1985) "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech", *Proceedings of the ICASSP 85*, March, 1985.

[87] Shieber, Stuart M. (1986) *An Introduction to Unification-Based Approaches to Grammar*. Center for the Study of Language and Information, Stanford, CA, 1986.

[88] Shieber, Stuart, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and Mabry Tyson (1983) "The Formalism and Implementation of PATR-II", in Grosz, Barbra J. and Mark E. Stickel *Research on Interactive Acquisition and Use of Knowledge: Final Report SRI Project 1894*, SRI International, Menlo Park, California, pp. 39–79.

[89] Sidner, Candace L. (1981) "Focusing for the Interpretation of Pronouns", *American Journal of Computational Linguistics*, 7.4, pp. 217–231.

[90] Sidner, Candace L. (1983) "Focusing in the Comprehension of Definite Anaphora", in M. Brady and R. C. Berwick, eds., *Computational Models of Discourse*, MIT Press, Cambridge, MA, pp. 267–330.

[91] Soong, F. and E. Huang (1990) "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition", in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*, Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 12–19.

[92] Soong, F. and E. Huang (1991) "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition", in *Proceedings of the ICASSP 91*, Toronto, Canada, pp. 705–708.

[93] Stallard, David (1989) "Unification-Based Semantic Interpretation in the BBN Spoken Language System", in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15–18, 1989*, Morgan Kaufmann, Publishers, Inc., San Mateo, California, pp. 39–46.

[94] Steele, Guy L., Jr. (1984) *Common LISP: The Language*, Digital Press, Digital Equipment Corporation.

[95] Steinbiss, V. (1989) "Sentence-Hypotheses Generation in a Continuous-Speech Recognition System," *Proc. of the European Conf. on Speech Communciation and Technology*, Paris, Sept. 1989, Vol. 2, pp. 51–54.

[96] Webber, Bonnie L. (1978) *A Formal Approach to Discourse Anaphora*, Report No. 3761, Bolt Beranek and Newman, Cambridge, MA.

[97] Webber, Bonnie L. (1983) "So What Can We Talk About Now?", in M. Brady and R. C. Berwick, eds., *Computational Models of Discourse*, MIT Press, Cambridge, MA, pp. 331–372.

[98] Woods, W. (1980) "Cascaded ATN Grammars", *American Journal of Computational Linguistics, January-March 1980, vol 6, no. 1*, Association for Computational Linguistics, pp. 1–12.

[99] Young, S. (1984) "Generating Multiple Solutions from Connected Word DP Recognition Algorithms". *Proc. of the Institute of Acoustics*, 1984, Vol. 6 Part 4, pp. 351–354.